Конспект лекций по курсу «Сети ЭВМ»
Часть 1
Сурков Д.А., Сурков К.А.

Введение.

Речь идет о сети, когда есть разделяемые ресурсы.

Сетевыми ресурсами могут быть:

1) дисковая память (файловое хранилище)
2) принтер
3) модем
4) процессор

В сети есть система адресации, в отличие от просто соединенных компьютеров. Система адресации позволяет идентифицировать компьютеры в сети. Сеть – это аппаратно-программный комплекс. В ней есть узлы (обработка информации) и соединения (передача информации). Узлом может быть не только компьютер (например, маршрутизатор для перенаправления потока информации).

Вычислительные сети делятся на:

1) глобальные
2) локальные

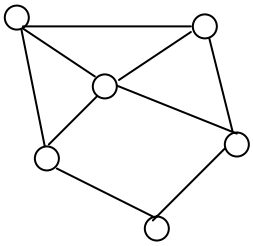
Вторые объединяют до несколько сотен узлов, первые – миллионы.

Отличие локальных от глобальных в принципе адресации. В локальных используется физическая адресация узлов. Адресом компьютера является адрес сетевого адаптера. В глобальных используется логическая адресация (IP-адрес).

В локальных сетях адрес компьютера уже имеется, компьютер может узнать своих соседей путем опроса.

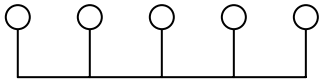
В глобальной сети нельзя узнать, кто есть в сети. Отсюда и отличия в протоколах.

Сеть:

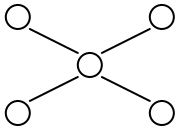


Сети еще классифицируют по другим признакам.
Сети отличаются по топологии (геометрии связи):

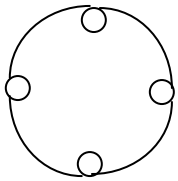
«шина»



«звезда»



«кольцо»



смешанная («шина», «кольцо»)

Бывают ячеистые (соединены, например, по три узла) и полносвязные (каждый узел связан с каждым).

Существует характеристика:

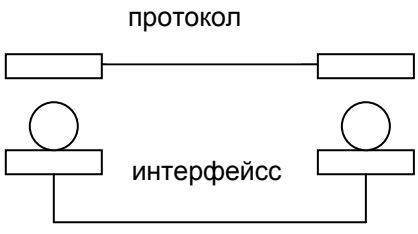
1) трафик в сети – количество информации, передаваемой во всей сети в единицу времени;
2) пропускная способность – количество единиц информации, передаваемой между двумя узлами в сети в единицу времени.

Единицами информации в сети обычно являются биты (т.к. физически информация передается битами), а также вводятся дополнительные служебные биты.

Чем больше трафик, тем меньше пропускная способность.

Протокол – язык общения узлов в сети.

Сетевой интерфейс:



Протокол – набор правил взаимодействия между адресатами (устройствами, находящимися на одном уровне сети).

Интерфейс – набор правил взаимодействия между уровнями сети.

Семиуровневая модель взаимодействия открытых систем

OSI – Open System Interconnection – семиуровневая модель:

7	прикладной	прикладной
6	представительный	представительный
5	сеансовый	сеансовый
4	транспортный	транспортный
3	сетевой	сетевой
2	канальный	канальный
1	физический	физический

Физический уровень отвечает за создание физической связи между узлами в сети. На нем стандартизируются параметры сигналов, способы кодирования сигналов, виды сигналов (потенциальный или импульсный).

Канальный уровень отвечает за преобразование байтов данных в последовательность битов. Он отвечает за разделение блоков данных на кадры. Кадры – маленькие порции данных, которые передаются по физическим каналам связи (линиям) и с помощью которых обеспечивается временное разделение общих физических линий связи между многими узлами. В один момент времени передается один кадр. Также канальный уровень отвечает за физическую адресацию.

Сетевой уровень отвечает за маршрутизацию в сети и обеспечивает передачу блоков данных между любыми двумя узлами. Также он обеспечивает прозрачное прохождение пакетов между сетями. На сетевом уровне обеспечивается интеграция нескольких сетей в одну. Этот уровень отвечает за логическую адресацию.

Транспортный уровень отвечает за надежную передачу пакетов или потока данных между узлами в сети. Он отвечает за разбиение блока (потока) данных на пакеты, передачу их по сети и сборку этих пакетов в правильном порядке. Также он отвечает за надежность доставки и должен обеспечивать повторную передачу при сбое.

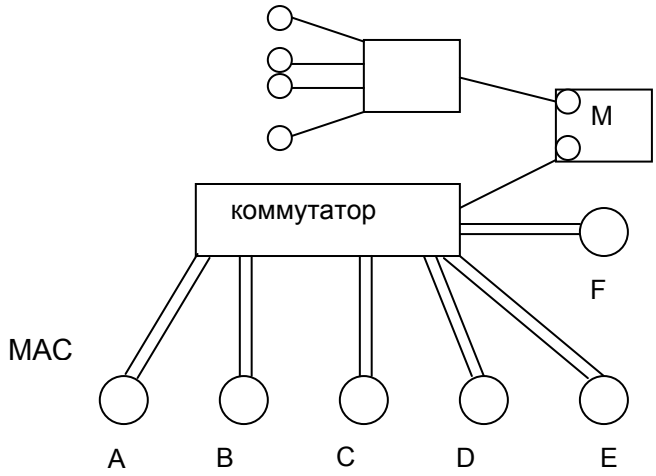
Сеансовый уровень отвечает за организацию диалога в сети, за установление контрольных точек, логического соединения и разрывов логического соединения.

Представительный уровень отвечает за преобразование данных из одного формата в другой, чтобы данные могли быть интерпретированы вышележащим уровнем (это надо при преобразовании в различные кодировки).

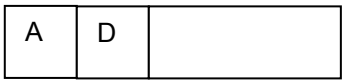
Прикладной уровень отвечает за предоставление конкретный прикладных услуг и реализацию различных сетевых служб (например, файловый сервер или служба удаленного доступа к терминалу, или служба доступа к Web-серверу).

Физический уровень – это провода.

На примере Ethernet:



Коммутатор просто ретранслирует сигнал. MAC-адрес есть у каждого компьютера. При передаче пакета указывается MAC-адрес отправителя и получателя. Каждый компьютер сразу читает MAC-адрес отправителя, потом – получателя и если адрес получателя совпал с его собственным, то принимает пакет.



Длина кадра всегда постоянна для данной сетевой технологии.

Сетевая плата байты преобразует в биты и передает их, а другая сетевая плата – биты в байты.

Канальный уровень – это сетевая плата.

Сетевой уровень: маршрутизатор обеспечивает прохождение некоторых пакетов из одной сети в другую. Если MAC-адрес не совпадает ни с одним MAC-адресом данной сети, то перенаправляется в другую сеть (это все анализирует маршрутизатор).

Сетевой уровень – соединение сетей и маршрутизация (тут работает IP-протокол).

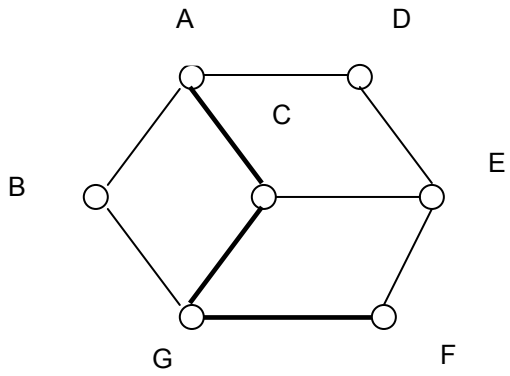
Транспортный: когда взаимодействуют разные программы на разных компьютерах. Работает TCP-протокол. Этот уровень отвечает за разбиение физического канала на логические каналы и надежную передачу данных.

Сеансовый уровень обеспечивает ведение диалога, установку соединения, разрыв, постановку контрольных точек.

Прикладной: все Интернет прикладные протоколы работают на этом уровне, серверы приложений, базы данных и т.д.

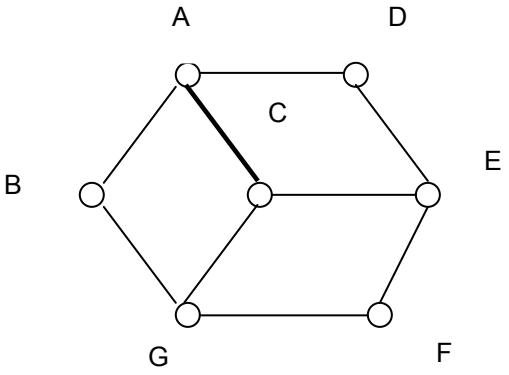
История развития сетей: вначале были большие машины с терминалами. Далее для связи с другими компьютерами начали использовать телефонные линии связи. Но сети потребовали несколько другой способ передачи данных, чем передача данных по телефону.

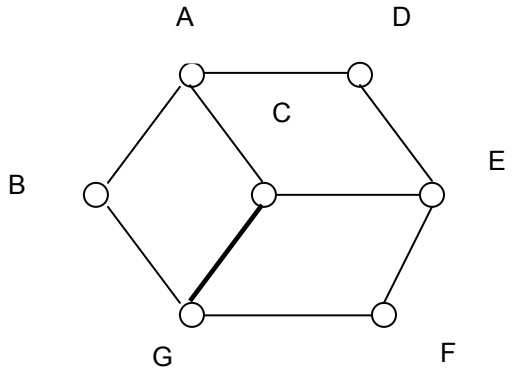
Рассмотрим компьютерную сеть:

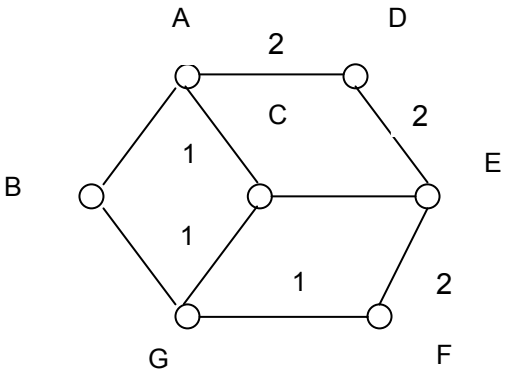


Если хотим сделать соединение A-F: после установки соединения скорость передачи информации высока, но каналы связи используются крайне неэффективно. Такая сеть называется сетью с коммутацией каналов.

Но придумали лучший способ: сеть с коммутацией пакетов – физический канал устанавливается не по всему маршруту (от начального узла к конечному), а к соседнему узлу. По этому соединению передается пакет данных, в который включается информация о конечном узле. Далее за передачу этого пакета отвечает узел, который его получил. После того как пакет передан, соединение может быть разорвано.





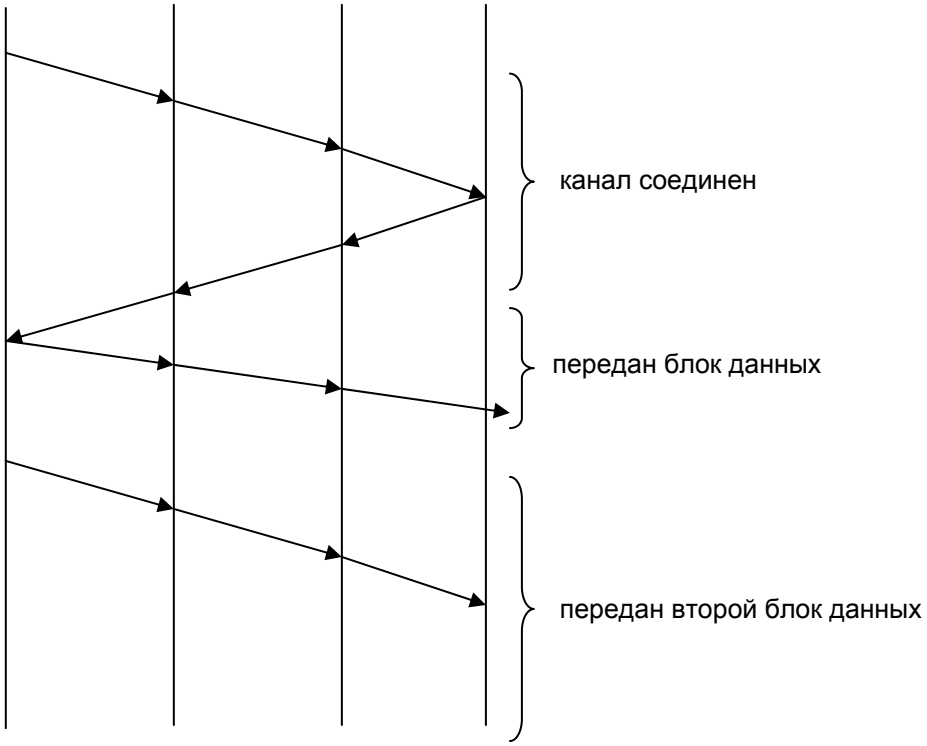


Пакеты могут одновременно (параллельно) передаваться и прийти к F различными путями

Сейчас сеть с коммутацией пакетов применяется не только в канальной связи (как раньше), а также и в IP-телефонии, когда сигнал (речь) модулируется, преобразовывается в цифровой вид, режется на пакеты и передается по сети. На конечном узле происходят обратные преобразования. Но эти мультимедиа данные занимают большой объем.

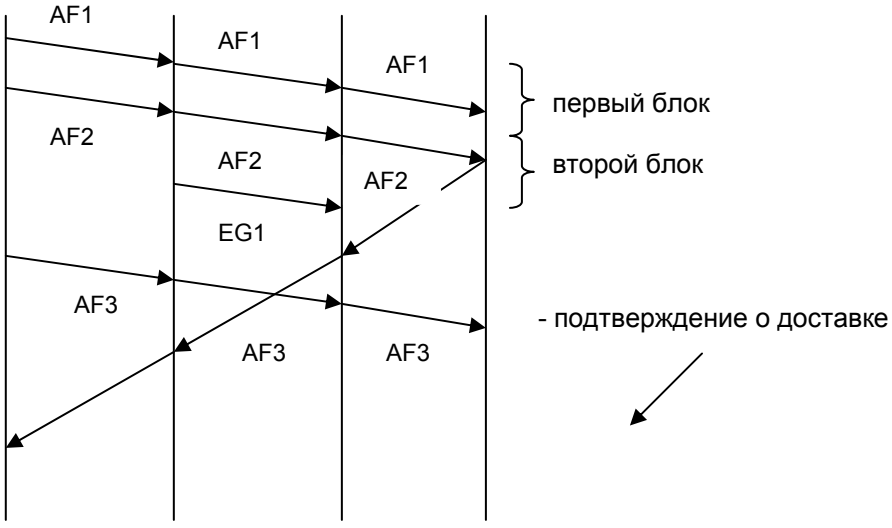
Временная диаграмма коммутации каналов:

A-C C-G G-F



С коммутацией пакетов:

A-C C-G G-F



Следующий блок передается без ожидания того, что получен предыдущий блок. Пакеты внутри себя идентифицируются.

Физический уровень.
Виды сред передачи данных. Кодирование сигналов.

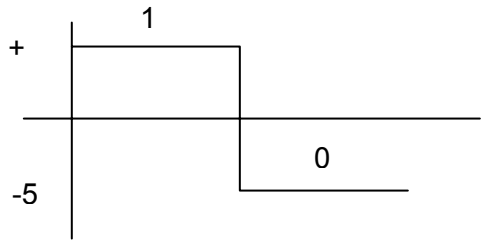
Для передачи сигнала из одного узла другому его надо закодировать. Кодирование – это способ представления информации. Информация в компьютере является двоичной, и надо выбрать способ представления сигналов в виде 0 и 1. Сигналы бывают разными:

1) аналоговые
2) цифровые

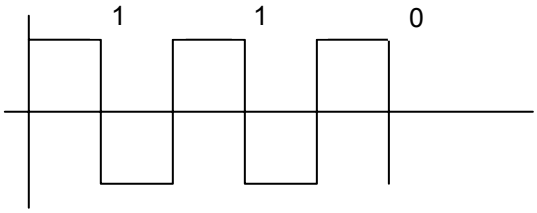
Аналоговый сигнал характеризуется непрерывно изменяющейся физической величиной (например, напряжением).

Цифровой (дискретный) сигнал характеризуется изменением какой-то физической величины между строго ограниченным числом значений. Для кодирования цифровых сигналов может применяться потенциальное/импульсное кодирование.

При потенциальном кодировании 0 и 1 характеризуется напряжением:

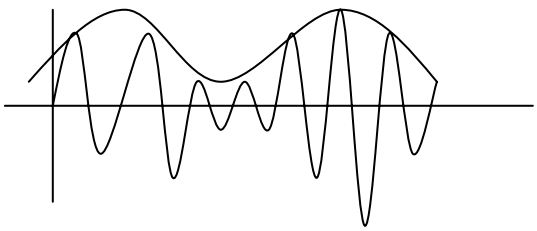


При импульсном кодировании 1 и 0 кодируются импульсами:

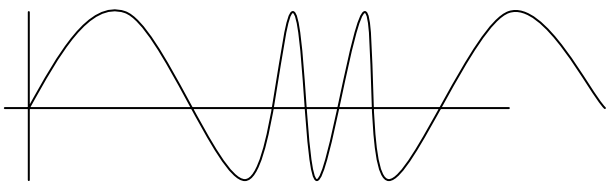


Для кодирования информации аналоговых сигналов используется модуляция (изменение какой-то характеристики сигнала по закону другого сигнала).

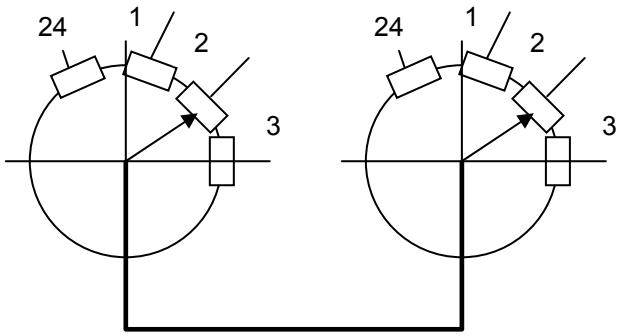
Есть амплитудная модуляция (используется в радиопередаче):



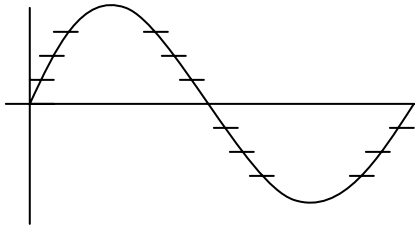
Частотная модуляция:



Фазово-импульсная модуляция (механическое представление):



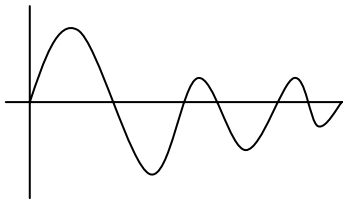
Стрелка замыкает контакт. Она вращается с очень большой скоростью. Сигналы передаются по кускам. Исключим сигналы (гармоники), которые меньше по частоте, чем две скорости вращения стрелки. Аналоговый сигнал превращается в цифровой. По одному физическому каналу связи в режиме разделения времени можем передавать много информации параллельно. Частота вращения стрелки – частота дискретизации.



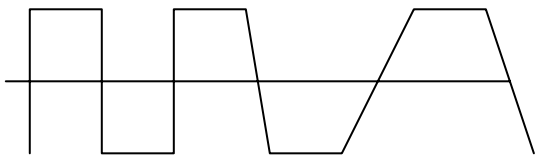
Теорема Котельникова: частота дискретизации должна быть как минимум в 2 раза больше, чем максимальная частота кодируемого сигнала.

При фазово-импульсной модуляции наблюдается высокая помехоустойчивость (т.е. устойчивость к наводкам).

При затухании аналоговых сигналов форма сигнала сохраняется, но уменьшается амплитуда:



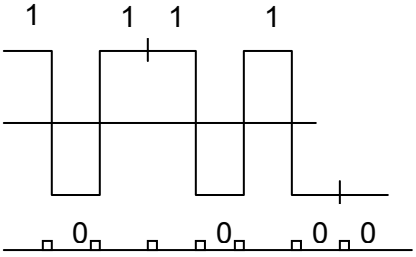
При затухании цифровых сигналов изменяется и форма:



Чтобы передать аналоговые сигналы на большие расстояния, используют усилители. При передаче цифровых сигналов используют повторители (распознает старый сигнал и предает такой же новый).

Способы кодирования сигналов:

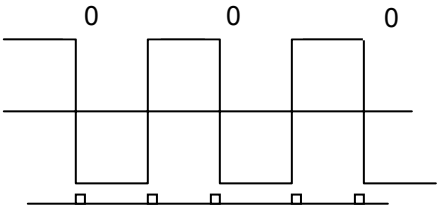
1) NRZ-кодирование (без возврата к нулю)



- синхроимпульсы

Для такого кодирования необходимо использовать синхронизатор (тактовый генератор) на обеих сторонах. Можно использовать либо два синхронизированных генератора, либо для синхронизации использовать дополнительную линию.

2) «Манчестерское кодирование»



Ноль кодируется переходом от положительной полярности к отрицательной, единица – от отрицательной к положительной.

Здесь также необходимо использовать тактовые генераторы (для случая 00 или 11), но в дополнительной линии нет необходимости.

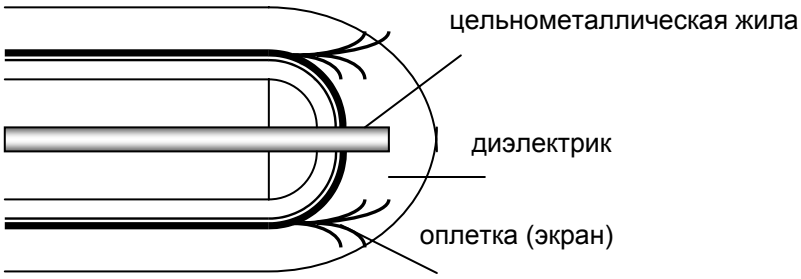
Физическая среда передачи данных и ее разновидности.

Виды физических линий связи (среды передачи данных):

1. Проводная среда
2. Беспроводная среда

Виды проводной среды:

1. Коаксиальный кабель:



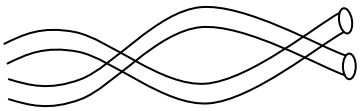
RG-58 – тонкий коаксиальный кабель. Существует толстый коаксиальный кабель, который позволяет увеличить длину сети, сохраняя пропускную способность:

тонкий – 25м

толстый – 500м

Чаще всего используется в топологии «шина». Скорость передачи 10Мбит/с.

2. Витая пара



Два свитых изолированных провода – идеальная форма для уменьшения влияния внешних помех. Для такой связи используют «манчестерское кодирование». Бывают 2-ух и 4-х жильные. Категорию витой пары определяет количество проводов, качество проводов, экранированность (используется металлическая оплетка). В зависимости от категории скорость может быть равна 4, 10, 100 Мбит/с, самые лучшие – 1 Гбит/с. Используется в топологии «звезда».

Передачи бывают:
1) симплексная (однонаправленная)
2) полудуплексная (передатчик и приемник находятся на двух сторонах, но передача происходит по одному каналу, разделение происходит по времени – когда один передает, другой принимает)
3) дуплексная (прием и передача происходит одновременно по двум направлениям с использованием двух каналов связи).

3. Оптические линии связи
Скорость передачи доходит до сотен Гбит/с.
Сравнение:
1 – удобно прокладывать, средняя стоимость, средняя пропускная способность
2 – низкая стоимость, выше средней пропускная способность, расстояние меньше (150 м)
3 – очень высокая пропускная способность, очень высокая стоимость изготовления и прокладки.

4. Радиосвязь, инфракрасная связь.

Сравнительная характеристика физических топологий.

Для локальных сетей применяется «шина», «звезда» и «кольцо».
«Шинная» топология:
1. Не требует дорогостоящих коммутаторов
2. Очень дешевая
3. Очень маленький расход проводов
4. При выходе из строя подключенного компьютера сеть продолжает работать
5. При удлинении «шины» требуется остановить работу сети
6. Низкая надежность при разрыве связи в проводах (но можно получить две сети, для чего надо поставить терминаторы).

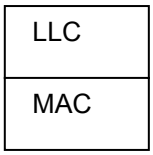
Топология «звезда»:
1. Наличие дорогого коммутатора, к которому подключаются все компьютеры
2. Высокая надежность, которая ограничена надежностью коммутатора; выход из строя сегмента сети не приводит к отключению сети
3. Большой расход кабеля

Топология «кольцо»:
1. Низкий расход кабеля
2. Отсутствие коммутатора
3. Высокая пропускная способность
4. Низкая надежность (разрыв кабеля ведет к разрыву сети)
5. Пропускная способность не зависит от трафика, т.е. трафик максимальный и максимальная пропускная способность.

Топологии бывают физические и логические.
Физические топологии – топологии линий связи.
Логические топологии – топологии логических каналов связи.

Канальный уровень.

Выделяют два подуровня:



MAC – Media Access Control – управление защитой данных
LLC – Logic Linking Control – управление логическими связями
Каждому компьютеру выделяется свой уникальный адрес (MAC-адрес).

Способы организации и функционирования логических топологий.

Методы доступа:

1. CSMA/CD – метод доступа с распознаванием несущей частоты, множественным доступом и обнаружением коллизий.

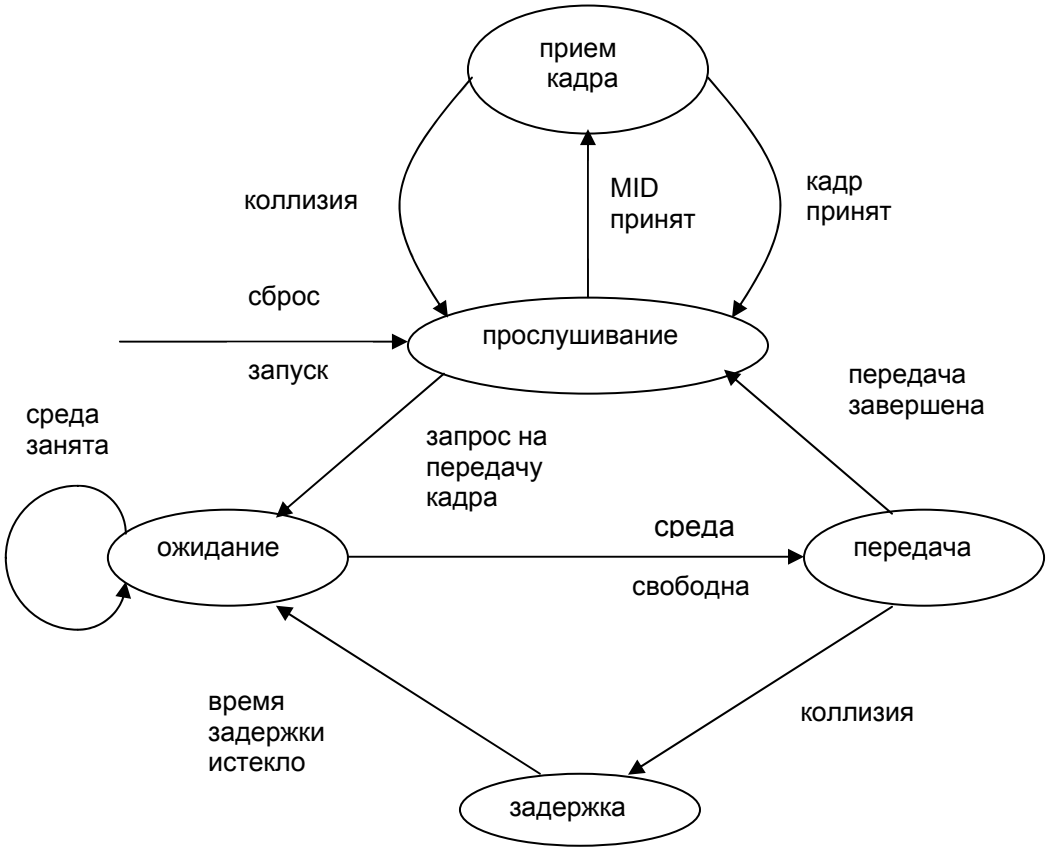
Все компьютеры слушают моноканал. Компьютер, который хочет что-то передать, должен дождаться, пока наступит тишина, и начинает передачу. Передача идет небольшими кадрами, в заголовке которых MAC-адрес целевого компьютера.

Может возникнуть ситуация, когда два компьютера хотят передать информацию, т.е. возникает коллизия. Чтобы этого избежать, компьютер слушает, что он передал, и, если это не совпадает с тем, что он хотел передать, он понимает, что возникла коллизия. Если компьютер обнаружил коллизию, то он штрафует себя на некоторое время. Это время уникально для каждого компьютера.

Есть два способа определения этого времени:

1) генератор случайных чисел
2) пропорционально MAC-адресу

Диаграмма перехода из состояния в состояние
(характеризует логику работы сетевой платы).



После запуска сетевая плата принимает MID (My ID). После приема кадра сетевая плата переходит в состояние прослушивания. Но может возникнуть коллизия, когда одновременно накладываются «передачи» от передающих станций. Из прослушивания переходит в ожидание по запросу на передачу кадра. Сетевая плата ожидает, пока канал не будет свободен (т.е. пока среда не станет свободной). Далее переходит в состояние «передача». После пожжет быть задержка или вновь прослушивание.

2. Модификация метода доступа CSMA/CD - CSMA/CA

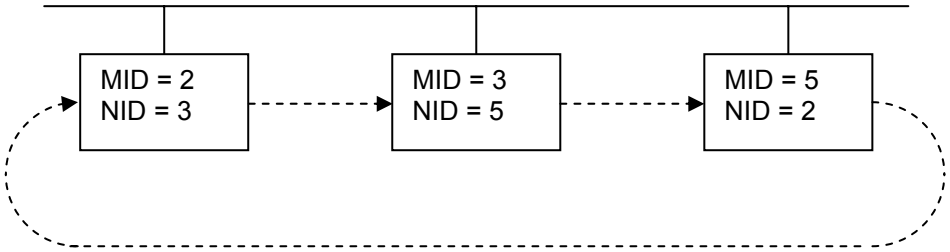
Кроме метода CSMA/CD (который используется в сетевой технологии Ethernet) есть модификация CSMA/CA – обнаружение несущей с множественным доступом и избеганием коллизий. Перед тем как передать кадр, сетевая плата передает короткий запрос на передачу – RTS-запрос (Request to send) – и только после этого начинает передачу кадра. Благодаря этому вероятность коллизий резко уменьшается, пакет RTS маленький и передается очень быстро. Вероятность столкновения RTS-запросов очень мала. Т.о., если компьютеры слышат, что кто-то передал RTS-запрос, то они свой RTS-запрос не передают, а ждут, пока будет передан кадр. Но могут быть столкновения RTS-запросов (тогда используется принцип, как и в CSMA/CD).

Метод CSMA/CA работает лучше при большом трафике, при небольшом трафике лучше работает CSMA/CD (в сети не циркулируют лишние RTS-запросы).

3. Физическая «шина» и логическое «кольцо»

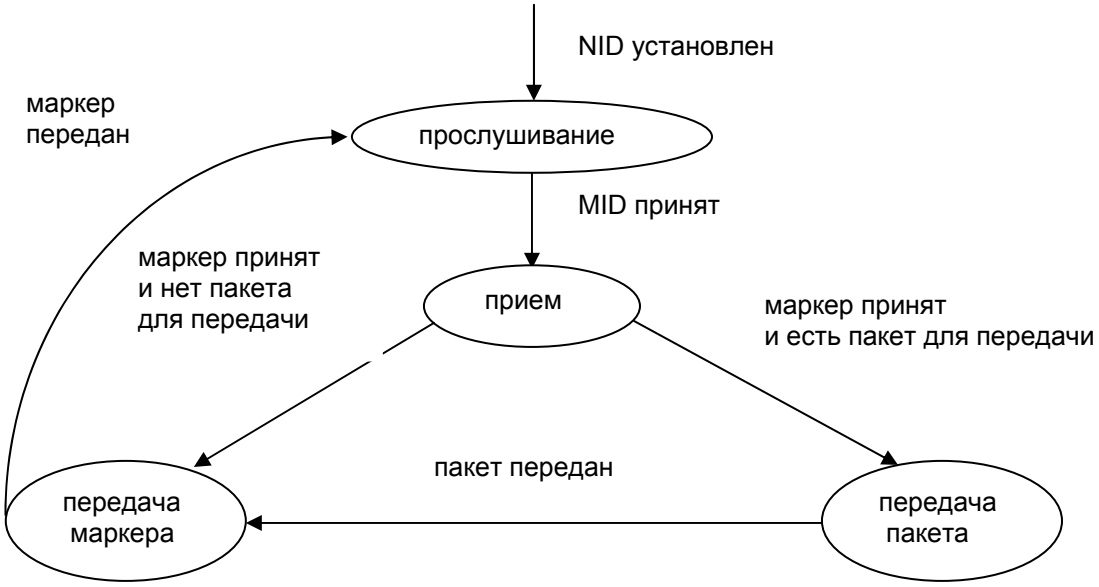
Метод с передачей маркера в топологии «шина».

Суть метода:



Физически используется шинная топология с передачей по моноканалу (т.е. все слышат, что передают другие). Но между компьютерами есть логическая последовательность перехода права на передачу. Право на передачу называется маркером. Каждый компьютер кроме собственного ID хранит ID следующего компьютера, которому передается маркер (NID – Next ID). Компьютер, который сейчас имеет маркер, передает пакет с маркером в сеть. Компьютер next, получив пакет от предыдущего компьютера, смотрит, пустой ли пакет. Если пакет не пустой, то он передается дальше (с маркером). Если пакет пустой, то этот компьютер «берет» маркер и начинает передачу.

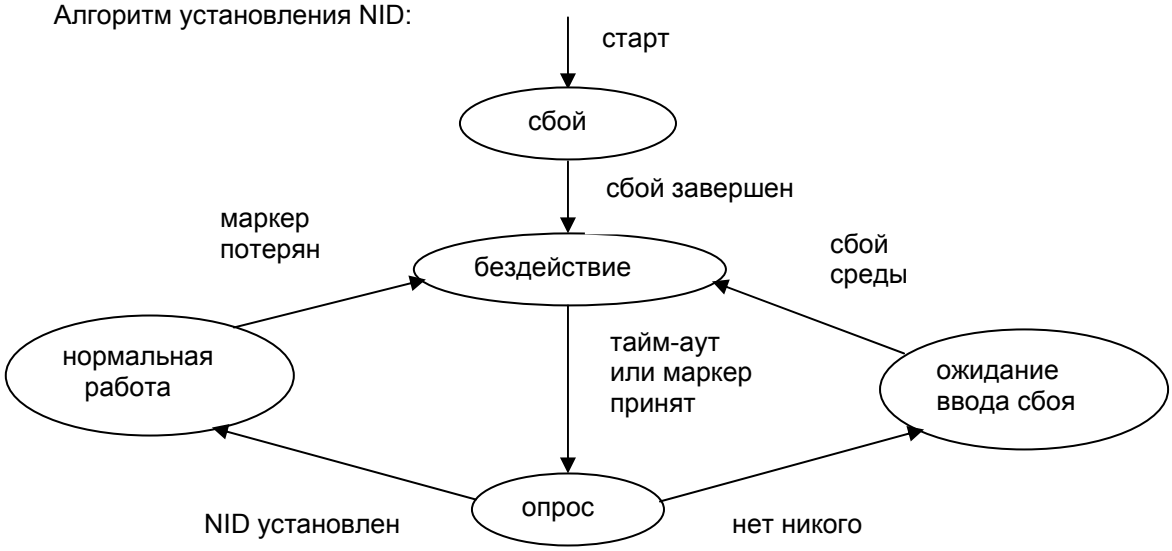
Схема состояний:



Когда NID установлен, компьютер переходит в состояние прослушивания. Далее он принимает свой собственный MID. Идет прием и просмотр на наличие пакета. После передачи пакета передается маркер. Когда передача маркера завершена, то переходим в состояние прослушивания.

Режим работы, когда сеть только включается или в нее входит новый компьютер.

Алгоритм установления NID:



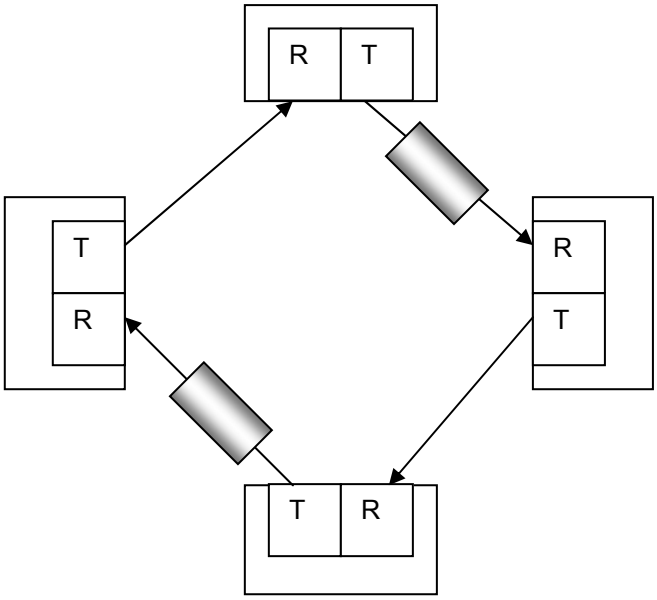
Общий случай: схема показывает, когда в работающую сеть включается новый компьютер, происходит сбой. Он посылает сбойную последовательность, которая приводит к потере маркера. Потом все компьютеры переходят в состояние бездействия. Там они находятся некоторое время (время тайм-аута). В этом состоянии компьютер ждет приема маркера. Время тайм-аута пропорционально MID (номеру сетевой платы). Тайм-аут у компьютера с меньшим номером истечет раньше. Когда время тайм-аута истекло, компьютер из состояния бездействия переходит в состояние опроса. В этом состоянии он начинает передавать маркер компьютеру с номером на 1 больше, чем у него. Компьютера с таким номером в сети может не быть. Поэтому после передачи маркера в сети будет тишина. В этом случае компьютер возобновляет передачу маркера компьютеру, номер которого больше на 2, чем его собственный и т.д. Если обнаруживается компьютер, у которого такой номер, то он принимает этот маркер и переходит из состояния бездействия в состояние опроса (сам начинает передачу маркера компьютеру с номером на 1 больше). А предыдущий компьютер, услышав, что маркер принят, запоминает номер компьютера в NID и переходит в нормальную работу.

Когда доходим до максимального значения MID, то счетчик переполняется и дальше устанавливается на 0. Далее находим компьютер с минимальным адресом. Когда он найден, все компьютеры переходят в нормальную работу (в режим прослушивания).

Переход из «опроса» в «ожидание ввода сбоя» означает, что компьютер в сети один (ждет, пока не появится еще один компьютер).

4. Логическая топология «кольцо»

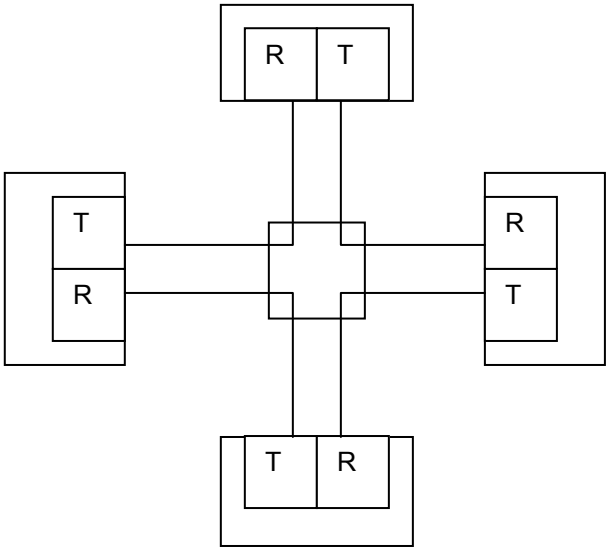
Когда логическая топология «кольцо» работает поверх физической топологии «кольцо», то это означает, что по кругу между компьютерами передается маркер.



R – receiver - приемник
T – transmitter – передатчик

По сети, по кольцу передается маркер. У компьютера физически есть соседний компьютер, от которого идет передача, и компьютер, которому идет передача. Если при передаче маркера у узла есть блок данных для передачи, то он передается вместе с маркером, иначе – только маркер. У маркеров нет адресов. Если станция принимает непустой пакет, но не адресованный ей, то пакет только ретранслируется. А та станция, которой адресован пакет, устанавливает признак того, что пакет получен, и шлет его дальше. Станция-отправитель получает этот пакет и делает вывод о корректности передачи. Потом она освободит маркер.

Физическая топология может отличаться от логической. Например, физической топологией может быть «звезда». Для удобства ставят коммутатор.



При работе с такой топологией в сети может существовать несколько маркеров одновременно. По сути, маркер постоянно циркулирует по сети, и пропускная способность мало зависит от трафика. Она максимальна. Скорость передачи маркера очень высокая.

Устройства канального и сетевого уровня.

1. Усилитель

Используется для удлинения линии связи аналогового сигнала с частотной модуляцией.

2. Повторитель

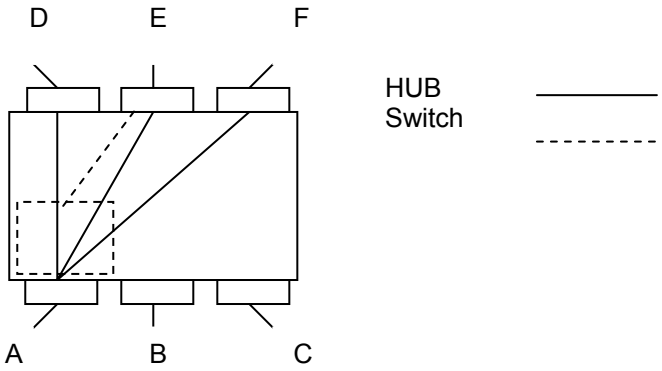
Используется для удлинения линий связи, в которых передается цифровой сигнал с одним из импульсных способов кодирования. Он принимает сигнал и потом просто повторяет его. Восстанавливает правильную форму.

3. Концентратор (HUB)

Используется для соединения линий связи. Обеспечивает ретрансляцию сигнала с одного входа на много выходов. Применяется в топологии «звезда». Работает на физическом уровне (это физическая коммутация проводов).

4. Коммутатор (Switch)

Служит тем же целям, что HUB, но в отличие от него Switch может дополнительно выполнять фильтрацию пакетов благодаря тому, что он может интерпретировать пакеты (кадры) сетевой технологии (например, Ethernet), т.е. знает Mac-адрес. Switch работает на канальном уровне сетевой технологии. Также Switch может осуществлять буферизацию.



Пакеты не могут быть подслушаны.

Остальные сети не будут загружены ненужными пакетами.

Позволяет передавать в топологии «шина» одновременно несколько кадров.

5. Маршрутизатор (Router)

Устройство сетевого уровня, которое обеспечивает не только коммутацию линий связи, но и маршрутизацию пакетов сетевого уровня. По своей логике работы и устройству оно схоже с Switch, но в отличие от него маршрутизатор распознает не кадры канального уровня (Ethernet), а пакеты сетевого уровня (IP-пакеты). Использует для фильтрации пакетов не физический адрес (MAC), а логический (IP). По сути, маршрутизатор – это целый компьютер, у которого столько сетевых плат, сколько у него портов. По каждому порту он может принять пакет, буферизирует его, а дальше обрабатывает эти пакеты по заложенному в нем алгоритме. Маршрутизатор имеет процессор. Они дорогие.

6. Мост (Bridge)

Устройство, которое служит для соединения 2-ух сетей. Мост работает на сетевом уровне. Это простейшая форма маршрутизатора (имеет 2 порта).

7. Шлюз (Gateway)

Это программа, которая обеспечивает передачу пакетов одного протокола внутри пакетов другого протокола и прозрачное соединение сетей, которые используют разные сетевые протоколы.

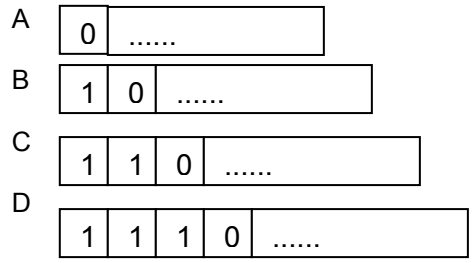
Сетевой уровень. Протокол IP.

Это протокол сетевого уровня, маршрутизируемый протокол, использует логическую адресацию. Логический адрес IP-протокола состоит из 4-х байт, которые записываются в виде 4-х десятичных чисел, разделенных точками:

192.168.1.2

IP-адреса используются в Internet. IP-адреса придуманы для уникальной адресации компьютеров в глобальной сети.

IP-адреса не имеют производителя, который их делает. В этом проблема. Поэтому была введена классовая адресация:



A: 1-ый байт – номер сети
2-4-ый – номер компьютера
B: 2:2
C: 3:1
D: зарезервирован для маршрутизатора

Это неэффективно. Поэтому была введена бесклассовая адресация, позволяющая делить сам адрес на адрес сети и подсети. Для этого вводится маска:

IP 192.168.234.100
Маска 255.255.255.0 (1111 1111.11… .0000 0000)
IP and Маска = номер узла
IP and (not Маска) = номер сети

В среде IP-адресов некоторые значения имеют особый смысл, они зарезервированы:
127.0.0.1 – означает сам узел. За этим номером обычно закреплено имя localhost.
255.255.255.255 – широковещательный адрес (все компьютеры в своей подсети).
0.0.0.0 – сам узел (при посылке сообщения посылка в стек протоколов не выполняется).

Для того чтобы работать в локальной сети достаточно установить IP-адрес, но для работы в глобальной сети необходимо задать:

1. Маску подсети
2. IP-адрес стандартного шлюза

Например:
IP: 192.168.234.100
Маска: 255.255.255.0
Gateway: 192.168.234.19
(192.168.234.0 – номер подсети)

1 -> Если посылается пакет по протоколу IP, то берется адрес, на который отправляется пакет, и на него накладывается маска. Т.о., выясняют номер узла и номер сети, в которую направляется пакет.

Если номер подсети совпадает с номером в своей подсети, то используется алгоритм передачи 1 , а если нет, то алгоритм 2.

2 -> Пакет отправляется своему стандартному шлюзу. Шлем пакет по адресу 192.168.235.20. Номер подсети – 192.168.235.0. Пакет будет направлен компьютеру 192.168.234.19, но адресат остается прежним.

Посылка пакета по протоколу IP выполняется с использованием 2-ух других протоколов:

1. ARP- address resolution protocol- протокол с преобразованием адресов. Преобразует IP-адрес в MAC-адрес.
2. RARP – reverse ARP- обратное преобразование адресов. Преобразует MAC-адрес в IP-адрес

ARP работает следующим образом: в сеть посылается широковещательный ARP-запрос, в который вкладывается IP-адрес искомого компьютера. Каждый компьютер, получивший запрос, сравнивает запрашиваемый IP-адрес со своим. Если адрес не совпадает, то запрос игнорируется, если совпадает, такой компьютер генерирует ARP-ответ и таким образом узнает MAC-адрес для соответствующего IP.

Протокол ARP используется при посылке в локальную сеть любых пакетов IP-протокола.

Проблема: при выполнении трансляции IP-адреса в MAC-адрес при каждой посылке сеть будет перегружена. Для решения используется кэширование соответствий IP-адресов MAC-адресам.

Каждый компьютер хранит таблицу соответствий IP и MAC-адресов и время выполнения данного запроса. Если время достаточно большое, запись стирается.

Чтобы по MAC-адресу узнать IP-адрес отправляется запрос на MAC-адрес.

RARP применяется для удаленной загрузки компьютеров в сети, т.е. для работы «бездисковых» компьютеров (дает экономию на жестком диске, т.к. механические устройства отстают). Компьютеру при удаленной загрузке необходимо узнать адрес сервера, с которого он загружается.

Формат IP-пакетов:

Заголовок

Версия	Длина заголовка версии	Тип службы	Общая длина в байтах	
Идентификатор пакета (16 бит)			Флаг 2 бита	Смещение фрагмента (14 бит)
Время жизни (8 бит)		Протокол (8 бит)	Контрольная сумма (16 бит)	
IP отправителя (32 бита)				
IP получателя (32 бита)				
Параметры			дополнение 0-ми до границы 4Б	
Данные				

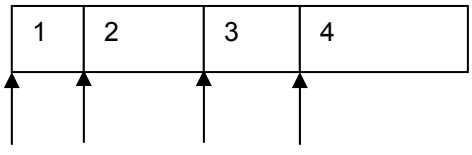
Версия задает версию протокола.
Длина заголовка в 32-х битных словах (отсчитывается от 5).
Тип службы определяет требования, которые налагаются на передачу пакета.

PR	D	T	

Биты управляют тем, как выбирается маршрут:
PR – приоритетная передача
D – минимизирована задержка
T – приоритет отдается пути с наименьшим трафиком
R – путь с максимальной надежностью.
Общая длина IP-пакета в байтах.
Идентификатор пакета – номер пакета, используется в тех случаях, если IP-датаграмма (data) была разбита на кусочки (собирает).
Флаги:

O	M

O - не фрагментировать
M – нет больше фрагментов
Смещение фрагмента – если IP-датаграмма фрагментирована, то это смещение в байтах фрагмента внутри самой датаграммы. Смещение по сравнению с индексом позволяет делить датаграмму на неравные фрагменты:



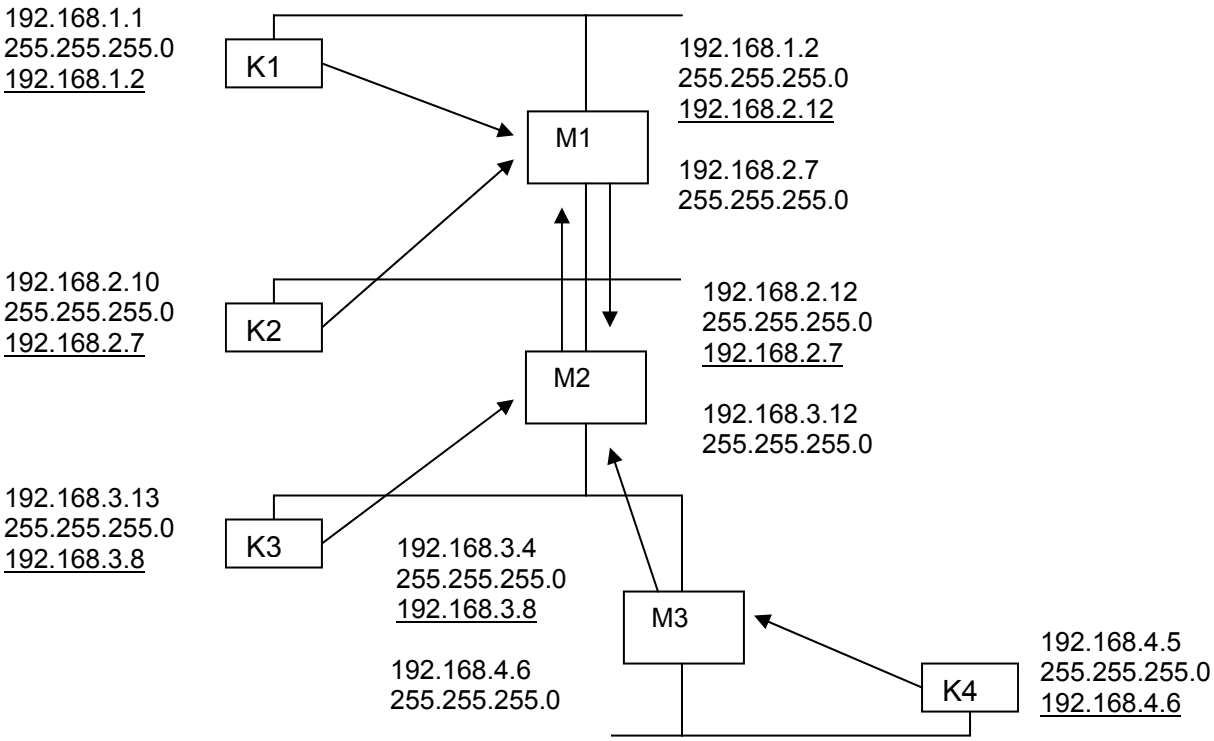
Время жизни – количество секунд, соответствующее времени нахождения пакета в сети, а затем количество маршрутизаций, которое должно быть выполнено (каждый маршрутизатор уменьшает его на 1). Когда оно равно нулю, пакет уничтожается. Помогает избежать зацикливания пакета в сети.

Протокол – номер (код) протокола более высокого уровня, которому отдается пришедший пакет.

Контрольная сумма – дополненная до 1 сумма всех 16-ти разрядных слов в заголовке (только заголовка).

Параметры – дополнительные параметры, с помощью которых можно расширять протокол IP различными командами. По формату их кодирование напоминает кодирование в процессоре Intel.

Конфигурирование сети. Соединение n сетей с помощью n-1 мостов.



При соединении n сетей n-1 мостами n ограничено 3. Необходимо использование маршрутизатора.

Для M2 если необходимо переправить:

192.168.1.0 -> 192.168.2.7 192.168.4.0 -> 192.168.3.4	} таблица маршрутизации

Критика протокола IP.

При проектировании протокола главное внимание уделялось надежности функционирования глобальной сети. Вопрос безопасности не рассматривался.

1. Если посылается ARP-запрос, шпионский компьютер может послать свой MAC-адрес.
2. При разработке сетевых протоколов разработчики не учли, что у разных компьютеров представление данных в памяти отличается. Если в двоичном виде мы будем посылать данные, то программа будет неправильно интерпретирована.
3. Данные, которые передаются в IP-протоколе, никак не типизированы. Следовательно, в каждой программе приходится использовать свою интерпретацию данных, эти данные очень трудно фильтровать и подменять (в заголовке все фиксировано).
4. Непродуман вопрос расширения протоколов. Количество протоколов ограничено. Если мы хотим создать свой протокол, не существует гарантированного способа понять, по какому протоколу происходит обмен сообщениями. Первый способ – закрепить номера портов. Но администратор сети может задать этот номер.

Протокол UDP.

Обеспечивает обмен датаграммами между процессами, работающими на компьютере.

В IP-протоколе не идентифицируются программы, а взаимодействуют именно они.

Придумана концепция портов – числовой номер (16-ти разрядный), который предназначен для того, чтобы отмечать закрепленную за ним программу.

Протокол UDP – протокол без установления соединения, без повторной передачи пакетов, без подтверждения передачи (без квитирования), без разбиения UDP датаграмм на пакеты.

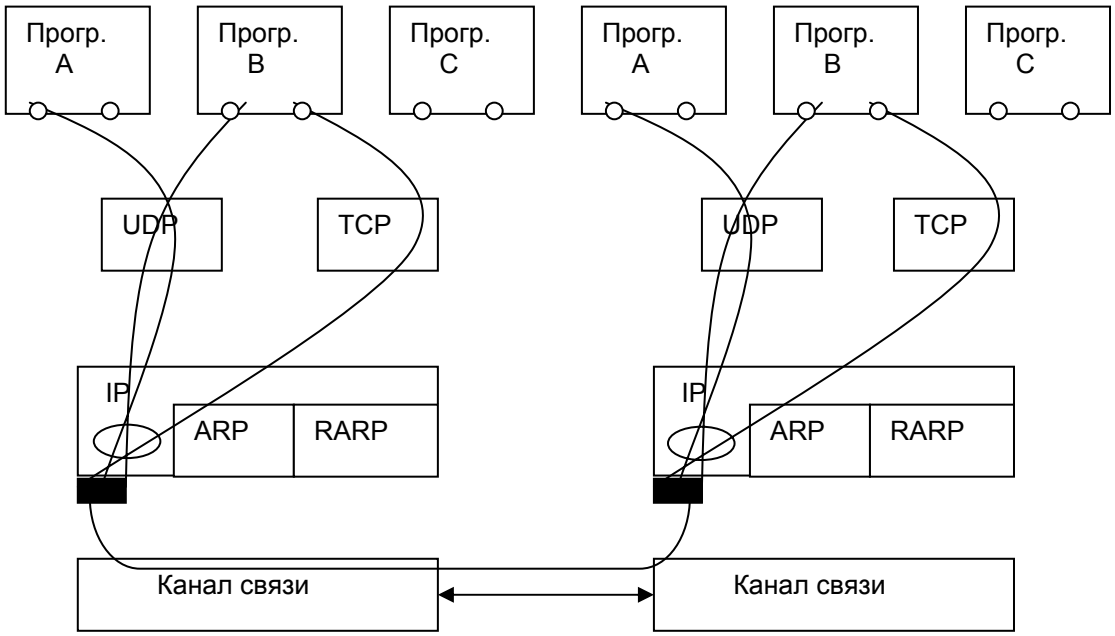
Формат UDP-пакета:

№ порта отправителя	№ порта получателя
Длина	Контр. сумма

Этот заголовок помещается перед IP-заголовком.

Порт служит для мультиплексирования каналов связи.

IP-адрес отправителя, IP-адрес получателя, порты отправителя и получателя являются уникальным идентификатором логического канала связи в сети.



На уровне IP происходит мультиплексирование протокола (например, TCP,UDP). Чтобы знать, какой протокол используется, для этого есть код протокола в поле IP-пакета. По коду протокола определяется, куда направляется пакет, который идет наверх (в протоколы более высокого уровня) – UDP или TCP. Драйвер протоколов UDP или TCP обеспечивает в свою очередь мультиплексирование канала связи. Для этого используется понятие порт.

Каждую линию уникально идентифицируют 4 цифры:

1) № порта отправителя
2) IP отправителя
3) № порта назначения
4) IP-адрес назначения

IP-пакет:

Протокол TCP.

Протокол с установкой соединения. Он гарантирует доставку данных и порядок получения данных. Протокол с квитированием, отправляемые данные всегда подтверждаются. Обеспечивается повторная передача данных в случае ошибки или если данные не дошли. TCP обеспечивает разбиение данных на фрагменты и сборку этих фрагментов прозрачно для пользователя.

UDP используется для передачи мультимедиа данных в реальном масштабе времени (потери не так важны, главное - скорость).

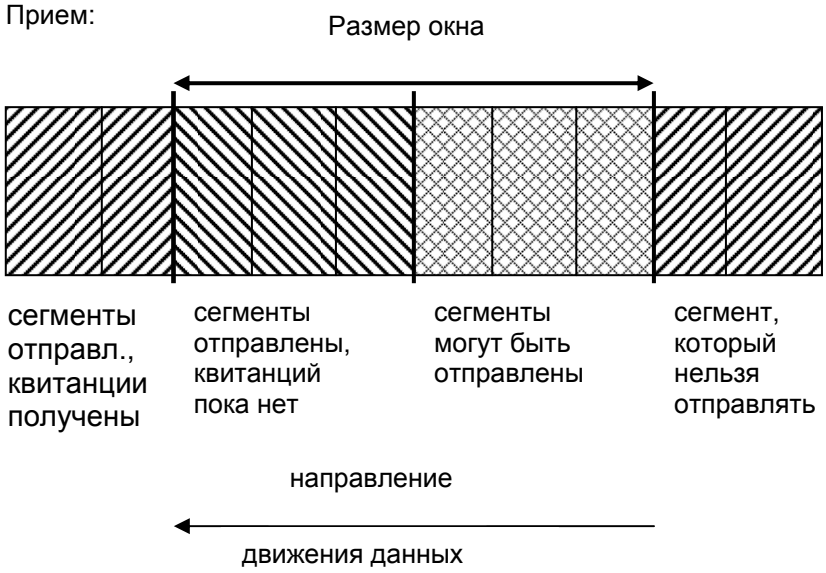
В основе работы TCP лежит принцип «скользящего окна».

Принцип: при передаче данных блоками, чтобы обеспечить надежность, необходимы подтверждения. Пусть на каждый блок будем посылать подтверждение. Трафик увеличится в 2 раза.

Выход: подтверждается не каждый блок данных.

Единицей подтверждения является байт. Пакеты могут быть любых размеров. Размер пакета может динамически меняться во время обмена данными, подстраиваясь к пропускной способность сети.

Прием:

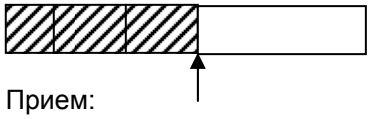


Это снимок «скользящего окна».

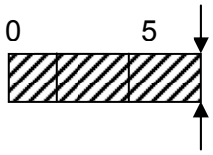
Размер окна определяет, какой объем данных может быть послан в отсутствие подтверждения. Размер окна может динамически меняться во время передачи данных. Чем больше размер окна, тем реже приходят подтверждения. Если в сети возможны потери данных или искажение, то тем больше вероятность, что между подтверждениями возникнет ошибка. Если она возникнет, то придется повторно передавать большой объем данных. Чем меньше размер окна, тем меньше объем данных, который необходимо повторно передавать в случае ошибки, но при этом больше общий трафик сети.

Вывод: размер окна нужно подстраивать динамически под пропускную способность сети. Протокол TCP адаптируется к трафику. Отрицательные квитанции не посылаются. Отсутствие квитанции означает либо прием искаженных данных, либо потерю квитанции. В качестве квитанции получатель сегмента посылает ответное сообщение, в которое помещает число, на 1 превышающее максимальный номер полученного байта.

Передача данных:



Прием:



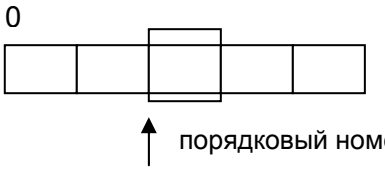
Отсылаем квитанцию 52, размер 32 бита.

Заголовок протокола TCP:

0 4 1 3

Порт отправителя			Порт получателя
Порядковый номер			
Номер подтверждения			
Длина загол.		Флаги	Размер окна
Контрольная сумма			Указатель срочных данных
Дополнительные параметры			Дополнение
Данные			

Порт отправителя и порт получателя – 16-ти разрядные имена портов.
Порядковый номер - № первого байта в поле «данные» TCP-фрагмента.



По нему определяется, куда вложить фрагмент в буфере при получении.

Номер подтверждения используется при отсылке TCP-пакетов, которые являются квитанциями. Если TCP-пакет является квитанцией, то поле «номер подтверждения» содержит № следующего байта, который ожидает получить получатель.

Длина заголовка – размер TCP-заголовка в 32-х разрядных словах.

4…10 биты зарезервированы.

Флаги могут иметь следующие значения:

FIN – завершение соединения (этот TCP-пакет – команда завершения)

SYN – для синхронизации порядковых номеров между адресами, означает установку соединения (устанавливается в 0)

RST – сброс соединения

PSH – проталкивание данных из приемного буфера (по возможности через все буферы)

ACK – acknowledgment – флаг подтверждения, т.е. если флаг установлен, то пакет является квитанцией

URG – urgent – передаются срочные данные

Размер окна содержит размер окна, позволяет во время передачи установить другой размер окна.

Контрольная сумма считается по IP и TCP заголовку.

Указатель срочных данных - № байта внутри данных, с которого начинаются срочные данные (смещение).

Дополнение до 32-х битов.

Алгоритм «скользящего окна».

Особенность работы: единицей передаваемых данных является сегмент.

Скользящее окно измеряется в байтах и может изменяться во время работы.

При установке TCP-соединения не ясно, какова пропускная способность канала и какой размер окна необходимо выбрать. Поэтому используют следующий алгоритм:

1. Размер окна устанавливается максимальным, а потом сокращается пор мере работы.
2. Формула определения размера окна: при каждой передаче пакета рассчитывается время оборота пакета, т.е. время путешествия пакета от одного адресата к другому и назад. Эта величина определяется постоянно. При посылке каждого сегмента это время замеряется отдельно.

3. Размер окна определяется как средневзвешенное по последним 10-ти накопленным значениям. Есть 10 значений. Им назначаются коэффициенты от 1 до 10 (самый большой коэффициент у последнего значения). Эти значения суммируются и вычисляется среднее значение, где наибольший вес будут иметь последние значения размера окна. Определяем, сколько данных можно передать за это время (объем переданных данных : время).

Так как это вычисление происходит постоянно, формула позволяет динамически адаптировать размер окна под пропускную способность.

Например, если пропускная способность резко уменьшается, объем данных, передаваемых без подтверждения, надо уменьшить.

Частные случаи: предположим, отправитель передает данные, а на стороне получателя данные в буфер TCP-драйвера приходят, но прикладная программа за этими данными не обращается. Происходит переполнение буфера.

Драйвер TCP поступает следующим образом: он начинает посылать подтверждения на последний принятый байт и устанавливает размер окна в 0. Это называется «зондирование пустым окном».

Результат: отправитель периодически получает подтверждение на один и тот же байт, т.к. он получает подтверждение (нет тишины в канале), отправитель знает, что пользователь еще «живой». Нулевой размер окна не позволяет ему еще отправлять данные, т.е. отправитель тоже останавливает передачу.

Гнезда (Socket).

Для работы с протоколом TCP и UDP имеется специальный программный интерфейс Socket API.

Для работы в сети используется концепция гнезд.

Гнездо – целочисленный номер, аналогичный по смыслу дескриптору файла.

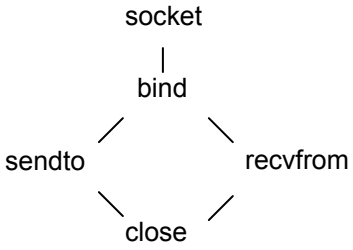
Создается гнездо, потом к нему привязывается IP-адрес или номер порта. Т.о., номер гнезда служит заменителем в программе IP-адреса и порта. Этот номер можно использовать в операциях чтения/записи аналогично файловому дескриптору.

Этот программный интерфейс существует во всех ОС.

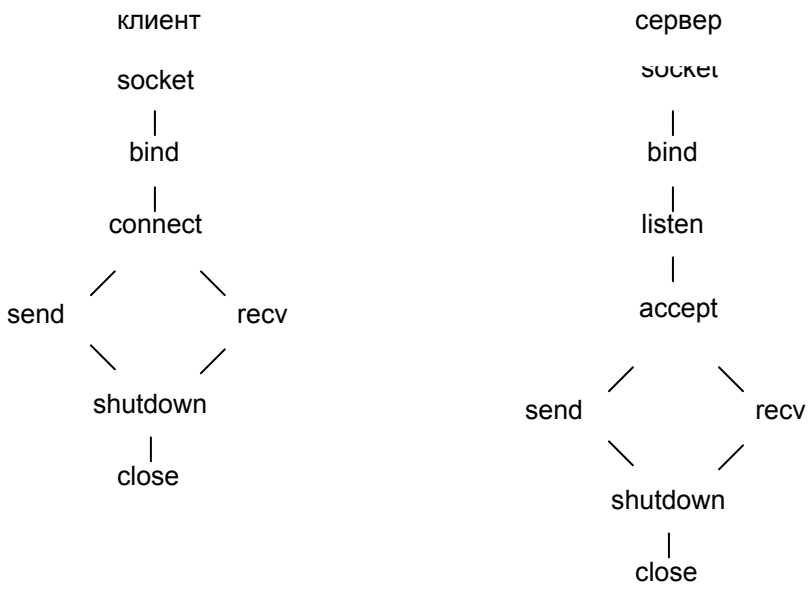
В Windows этот интерфейс был расширен, чтобы использовать поверх других протоколов. Дополнительно существует 2-ая версия WindSocket. В версии 2 поддерживается создание драйверов для любых новых протоколов.

Протоколы TCP/IP, NetBios также поддержаны в виде драйвера.

Схема при использовании UDP:



Протокол TCP:



Функция *socket* создает гнездо:
int socket (int af, int type, int protocol)
af: *AF_UNIX*
AF_INET
type: *sock_stream*
sock_dram

protocol: PROTO_TCP
PROTO_UDP
PROTO_XXX

```
struct sockaddr_in
{
   short sin_family;
   u_short sin_port;
   struct in_addr sin_addr;      // IP
   char sin_zero [8];            // дополнение нулями
}
struct sockaddr_4
{
   short su_family;      // семейство адресов
   char su_name [13];   //   имя host'a UNIX'a
}
union sockaddr
{
   sockaddr_in sin;   // для Internet
   sockaddr_u su;     // для Unix
};
```

Функция *socket* возвращает номер гнезда в качестве результата. Этот идентификатор служит идентификатором логической связи между компьютерами.
Параметр *af* –address family –семейство адресов
AF_UNIX – используется идентификация компьютера с помощью символьного имени
AF_INET – используется система адресов Internet'a
type – тип гнезда
для TCP: *sock_stream* – потоковое гнездо для передачи непрерывного потока данных
для UDP: *sock_dgram* – для посылки датаграмм
protocol – какой протокол будет использоваться. Константы помечают не только TCP и UDP, но и WindSocket (используется префикс WSA).

Структуры данных.
Чтобы назначить гнезду «номер», служит функция *bind*:
int bind (int socket, const struct sockaddr name, int namelen)*

Связывает с гнездом IP-адрес и порт, указанные в параметре *name*. В параметре *namelen* – размер структуры *sockaddr*.

По размеру можно проверить, корректно ли передана структура (той ли версии).

Структура *sockaddr* – это объединение структур. Размер структуры – максимум из размеров объединяемых структур. Возвращает 0 или код ошибки.

int connect (int socket, const struct sockaddr name, int namelen)*

Используется клиентом для установки соединения. Принимает номер гнезда, указатель на структуру с IP-адресом и портом.

int listen (int socket, int backlog)

Используется сервером. Переводит гнездо в режим прослушивания и организует входную очередь для приема запроса на установку соединения. Она не блокирующая.

Параметр *backlog* – длина очереди. Это внутренняя очередь на установку соединения. Если *backlog* = 5 и приходят 5 запросов, все попадут в очередь. Если 6, один не попадет.

int accept (int socket, struct sockaddr addr, int addrlen);*

Изымает из очереди запросов очередной запрос и обрабатывает его, создавая новое гнездо и возвращая его номер, IP-адрес и порт этого гнезда в структуре *sockaddr.*

int shutdown (int socket, int how)

Выполняет закрытие входного и/или выходного канала связи (задается полем *how*):

SD_SEND

SD_RECEIVE

SD_BOTH

Соединение TCP – shutdown (для UDP не используется).

При взаимодействии сервера с клиентом устанавливается дуплексная связь.

int send (int socket, const char buf, int len, int flags)*

int resv (int socket, char buf, int len, int flags)*

int sendto (int socket, const char buf, int len, int flags, const struct sockaddr* to, int tolen)*

int resvfrom (int socket, char buf, int len, int flags, const struct sockaddr* from , int fromlen)*

Функции *send, resv* используются в протоколах TCP и UDP.

s*endto* и *resvfrom* – только в UDP.

socket – номер сокета

buf - указатель на буфер данных, из которого или в который

len – длина буфера

flags – флаги:

MSG_OOB - out of band – срочные данные

MSG_XXX

sockaddr – непосредственно указываются IP-адрес и порт куда (to)/откуда (from) отправляются данные

Recv, recvfrom возвращают реальное количество байт. Когда вызывается функция recv, а канал данных уже закрыт, функция возвращает 0. В случае ошибки возвращает отрицательное значение. Т.о. отсутствует понятие EndOfFile, т.е. нельзя проверить соединение на то, что данные получены. Единственный способ узнать о конце данных - это принять их. В функции recv совмещены две функции: прием данных и проверка на конец.

int closesocket (int socket)

Закрывает гнездо, его дескриптор освобождается для повторного использования. Система выжидает некоторое время перед повторным использованием. Это связано с особенностью работы TCP.

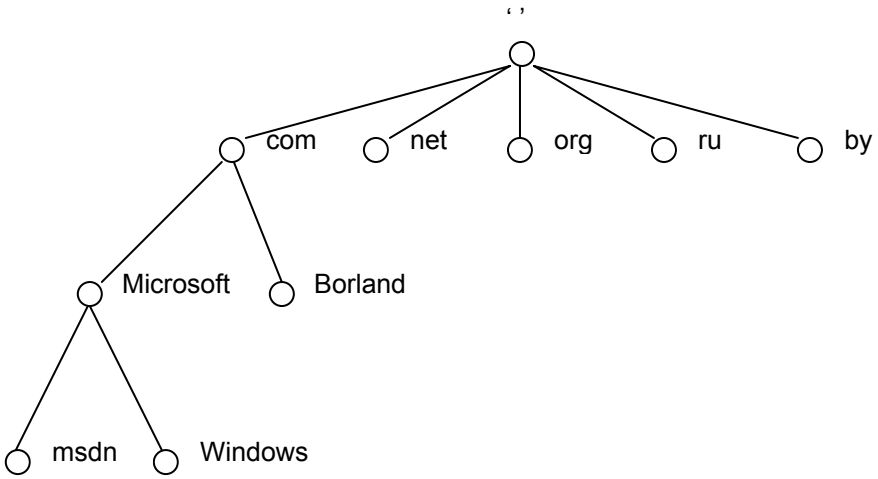
Система доменных имен DNS

Система DNS стандартизирует форматы имен и запросы на получение каких-то данных, ассоциирующихся с этими именами. Все имена объединяются в иерархию, представляющую собой дерево.

Имена обладают структурой, они составные.

Корнем дерева имен является пустое имя, имя без единого символа.

Добавляя к нему через точку префиксы, мы формируем уточненное имя, соответствующее некоторому узлу в глобальном дереве имен.



.com – коммерческии
.net – развитие сети
.org – организации
.mil – военный
.biz – бизнес
.by
.ru
.ua
.uk
} для стран

Каждый узел называют доменом.

Корень – нулевой уровень.

Имена, или домены первого уровня стандартизированы. Добавлением имен первого уровня занимается одна глобальная организация. На настоящее время они уже не добавляются.

Дерево можно нарезать на зоны. Зона отдается организации, которая занимается ее расширением. Делается это для того, чтобы куски дерева могли находиться на разных компьютерах.

msdn.microsoft.com

К этому имени могут быть подключены записи различного типа (IP, email).

- A (address) – записи об IP-адресах

DNS используется для хранения IP-адресов.

Рассмотрим, как выполняются DNS-запросы.

Они бывают двух видов:

- итеративные
- рекурсивные

Итеративный запрос.

На каждом компьютере, который подключен к Internet и DNS, имеется DNS-агент. Он ищет в своей локальной таблице соответствий искомое имя. Если нет, то он обращается с запросом к своему DNS-серверу верхнего уровня. DNS-сервер ищет у себя и возвращает либо результат, либо признак «не найден » + адрес другого DNS-сервера, который рекомендуется для продолжения DNS-запроса. Таким образом, в этом режиме DNS-агент выполняет перебор DNS-серверов сам, пока не будет найден ответ.

Рассмотрим пример:

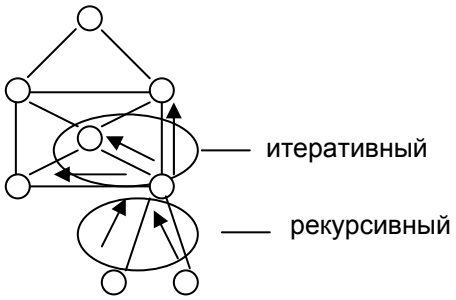
DNS-агент попытается обратиться к серверу .com, передав ему имя msdn.microsoft.com. Если в домене верхнего уровня этот адрес явно не прописан (а он не прописан), сервер возвратит «не

знаю» + IP-адрес сервера microsoft.com. DNS-агент обратится к microsoft.com, опросит его. На этом сервере хранится IP-адрес того DNS-сервера, на котором было зарегистрировано это имя. Возвратит «не найдено» + IP msdn.microsoft.com.

Перебор по циклу выполняет DNS-агент.

Рекурсивный запрос.

Заключается в том, что DNS-агент спрашивает DNS-сервер: «Узнай мне IP такого имени». DNS-сервер сам занимается поиском и возвращает конечный результат. Имеет место делегирование обязанности.



Технология Proxy.

Proxy- технология обеспечивает анонимность компьютеров конечных пользователей при работе с серверами Internet'а. Кроме того она обеспечивает большую безопасность, отгораживает локальную сеть от сети Internet, не позволяя подключаться к компьютерам конечных пользователей (находящихся в локальной сети) из сети Internet.

Задачи:

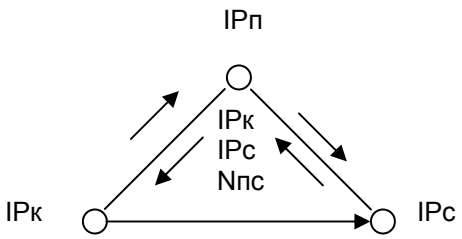
1. Не позволить серверу идентифицировать IP-адрес клиента
2. Не позволяет серверу подключаться к клиенту

Это осуществляется следующим образом:

На компьютере в локальной сети выделяется сервер, отвечающий за выход в Internet, и на него устанавливается Proxy-сервер. Клиентское ПО должно поддерживать концепцию Proxy-сервера.

Идея: когда клиент желает подключиться к удаленному серверу, он выполняет запрос к Proxy-серверу, сообщая ему IP-адрес и порт удаленного сервера, с которым он бы хотел работать.

Proxy-сервер делает следующее: он создает у себя сервер, который прослушивает указанный порт. Кроме этого он выполняет запрос на удаленный сервер, устанавливая с ним соединение (на этот же порт). Далее клиент выполняет запрос на установку соединения к Proxy-серверу на этот порт. При посылке пакетов клиентом Proxy-сервер принимает их и отсылает удаленному серверу от своего имени, подменяя в пакете исходящий IP-адрес. Когда данные идут от удаленного сервера к клиенту, они реально идут на IP-адрес Proxy-сервера, который отсылает их клиенту, подменяя в пакете IP-адрес удаленного сервера на свой адрес.



Клиент -> Proxy

IPк
IPп
Nпк/Nпп

Proxy -> Сервер

IPп
IPс
Nпп/Nпс

Сервер -> Proxy

IPс
IPп
Nпс/Nпп

Proxy -> Клиент

IPп
IPк
Nпп/Nпк

Эта технология называется Network Address Translation (NAT).

Клиент должен уметь формировать запрос Proxy-серверу с данными сервера, с которым он хочет общаться.

Протокол, который служит для того, чтобы запрограммировать Proxy-сервер на правильную трансляцию пакетов, называется Socks 4,5.

Бывают Proxy-серверы без использования Socks, т.е. программируемые вручную. Они могут просто выполнять соединение на другой порт без трансляции адресов. Proxy-серверы могут поддерживать конкретные протоколы, например, HTTP. Мы рассмотрели универсальный Proxy-сервер. Можно написать Proxy-сервер самим.

Технология передачи данных в сети

1. *FileStream fs = FileSrteam (“myfile.txt”)*
   int a = 65535; // 0x0000FFFF
   fs.WriteInt (a);

   Big-endian

0	0	F	F

   Little-endian

F	F	0	0

Компьютеры с разной архитектурой по-разному хранят числовые данные. Поэтому для передачи данных между компьютерами по сети их представляют в виде строк.

2. Изначально работа со многими протоколами выполнялась с помощью консольных программ-утилит. Пользователь вводил команды в текстовом виде. Результат также выдавался на экран, т.е. должен был представлен в текстовом виде.

Таким образом, исторически многие протоколы работают в текстовом виде.

Но есть возможность передавать двоичные данные по каналу таким образом, чтобы они корректно интерпретировались и компьютерами с разной архитектуры и разной разрядностью.

Идея: стандарт Oasis – формат обмена графической информацией.

Достоинства: данные хранятся компактно, позволяет записывать/читать двоичные данные любыми компьютерами, передача по сети.

Пусть мы записываем число int

0x0000FFFF

Байты изымаются из числа и пишутся последовательно. Последовательность записи: пишется младший байт, если следующий байт имеет значащие биты, то пишется следующим и т.д.

Данные записываются по 7 бит. Число разбивается на 7-битные фрагменты. К ним добавляется 8-ой, который показывает, есть ли еще в числе значащие биты (1-есть, 0-нет).

Запись идет побайтно, но из числа берем 7 бит.

Например:
8=00001000

0000100

32-х разрядное число будет записано в 1 байт.
258=100000010

1000001	0000001

Потребовалось 2 байта.

Чтение из потока:

```
#define SIGNIFICANT_BITS 7
#define CONTINUATION_BIT_MASK (1<<7)

template <typename T>
void ReadUnsignedInteger (Stream*  stream, T*  value)
{
    *value = 0;
    unsigned char b;
    int shift = 0;
    do
    {
        stream -> ReadByte (&b);
        b|=((b&(~ CONTINUATION_BIT_MASK))<<shift);
        shift+= SIGNIFICANT_BITS;
    }
    while ((b& CONTINUATION_BIT_MASK)!=0)
}
```

Протокол Telnet

Обеспечивает работу с удаленным компьютером в консольном режиме. При этом клиентский компьютер, по сути, выступает в роли консоли (текстовой), т.е. клавиатура + дисплей. Мощность самого компьютера используется для вводимых с клавиатуры символов и отображение на дисплей.

Все команды выполняются на удаленном сервере. При запуске программы Telnet-пользователь получает возможность выполнять несколько команд: подключение/отключение к/от удаленного сервера (подключение – open, отключение - close). В команде указывается имя или IP-адрес сервера.

Протокол FTP

Обеспечивает передачу и прием файлов на удаленный сервер и с удаленного сервера. За каждым таким протоколом закреплен номер порта (стандартом), и когда пользователю нужна служба FTP, используется порт №21.

Команды FTP обеспечивают следующее:

1) подключение к удаленному серверу, отключение (open, close)
2) get, put

get обеспечивает передачу файла с сервера на локальный компьютер
put – с локального компьютера на сервер

3) перемещение по директориям сервера и по директориям на локальном компьютере

dir – получение списка каталогов сервера
mdir – на своем компьютере

Протоколы передачи электронных сообщений SMTP, POP3

SMTP предназначен для передачи электронных сообщений. Клиент создает TCP-соединение с сервером через порт №25. Затем клиент обменивается с сервером сообщениями до тех пор, пока соединение не будет прервано. Через протокол SMTP клиент сообщает, какую операцию он хочет выполнить. Команда, которую он передает, представляет собой текст и состоит из ключевых слов, за ключевыми словами следует параметр через пробел. Каждая команда заканчивается парой символов: конец строки и перевод каретки (CRLF).

Синтаксис команд SMTP:
HELO <SP> <domain> <CRLF>
MAIL <SP> FROM: <reverse_path> <CRLF>
RCPT <SP> TO: <forward_path> <CRLF>
DATA <CRLF>
RSET <CRLF>
SEND <SP> FROM: <reverse_path> <CRLF>
SOML <SP> FROM: <reverse_path> <CRLF>
SAML <SP> FROM: <reverse_path> <CRLF>
VRFY <SP> <string> <CRLF>
EXPN <SP> <string> <CRLF>
HELP <SP> <string> <CRLF>
QUIT <CRLF>
NOOP <CRLF>

Пример: подключаемся к серверу
C: HELO 195.161.101.33
S: 250 smtp.mail.ru is ready

C: MAIL FROM: <dima@mail.ru>
S: 250 OK

C: RCPT TO: <kirill@mail.ru>
S: 250 OK

C: DATA
S: 354 start mail input; end with <CRLF>. <CRLF>

C: FROM: Dima <dima@mail.ru>
C: TO: Kirka <kirill@mail.ru>
C: Subject: Lecture
C: Hello Kirka! Your lecture is gone. You were late.
C: Students are waiting for you for 2 hours…
C: ↓
C: .↓
S: 250 OK

C: QUIT
S: 221 smtp.mail.ru is closing transmission channel

Протокол POP3 (Post Office Protocol).

Предназначен для работы с удаленным почтовым ящиком.

Клиент начинает работать с установки TCP-соединения на порт №110. На этом порту должен быть сервер, который прослушивает соединение. Когда соединение установлено, сервер посылает клиенту приглашение. После этого клиент и сервер обмениваются информацией, пока соединение не будет закрыто или прервано.

Клиент посылает команды, состоящие из ключевых слов, и через пробел идут аргументы. Длина ключевых слов – 3-4 символа. Длина аргументов – не более 40 каждый. Каждая команда должна завершаться символами CRLF. Ответы на некоторые команды могут состоять из

нескольких строк. В этом случае каждая строка разделена символом CRLF, а весь ответ - точкой и CRLF.

Когда отвечает сервер, команды бывают в двух формах:

+ OK текст – успешно

- ERR текст – ошибка

Основные команды:

C: USER Dmitry
S: +OK Dmitry is real user

C: PASS Surkov
S: +OK Dmitry's maildrop has 2 messages

C: STAT
S: +OK 2 320

C: LIST
S: +OK 2 messages (320 octets)
S: 1 120
S: 2 200
S: .

C: RETR 1
S: +OK 120 octets
S:
S: .

C: DELE 1
S: +OK message 1 deleted

C: RETR 2
S: +OK 200 octets
S:
S: .

C: DELE 2
S: +OK message 2 deleted

C: QUIT
S: +OK desk POP3 server signing off

C: <закрыть TCP-соединение>

USER передает серверу имя пользователя. Сервер проверяет синтаксическую правильность логина.

PASS передает пароль пользователя. Если пользователь есть, пароль подошел, ящик не заблокирован другими соединениями, значит +OK.

STAT – статистика, выдает количество сообщений в ящике и их размер, помеченные для удаления не учитывает.

LIST выдает информацию о сообщении с указанным номером.

DELE удаляет сообщение с указанным номером.

RSET сбрасывает сообщение, помеченное к удалению, делая его непомеченным.

RETR извлекает содержание сообщения с указанным номером.

QUIT переводит POP3-сервер в состояние update (обновления). В этом режиме сервер удаляет сообщения, помеченные к удалению, и завершает POP3 сессию.

TOP выдает заголовки сообщений (сообщения с указанным номером), выдаются первые n строк.

Протокол HTTP

Это протокол передачи гипертекстовых документов. Этот протокол используется браузерами.

В нем используются следующие команды:

1) открытие соединения
2) Get запрашивает документ
3) Post посылает документ на сервер
4) закрытие соединения

Было введено понятие URL (Uniform Resource Local) – универсальный идентификатор ресурсов. Имеется следующий формат (в общем виде, не только для http):

<имя протокола> // <имя пользователя>:<пароль>@<имя узла>:<имя порта>/<имя ресурса>

Пример:

ftp:// ds:sim-sim@ftp.borland.com:2121/download/Delphi

http:// …

http:// www.google.com

Основы RPC

Remote Procedure Call – удаленный вызов процедур.

Идея RPC: кроме обмена данными между программами в их взаимодействии бывает нужна передача управления, т.е. вызов программы на другом компьютере или выполнение какой-то услуги, сервиса.

Часто требуется при программировании перенести библиотеку подпрограмм на удаленный компьютер и обеспечить возможность вызова этих подпрограмм с клиентских компьютеров.

Например, есть функция вычисления 120 знаков числа ПИ. Мощности может не хватать, поэтому пусть эти действия производит другой компьютер, на него перенесем библиотеку.

Идея RPC: обеспечить прозрачное использование удаленной библиотеки подпрограмм так, как будто она является частью прикладной программы на локальном компьютере.

.h void GetAlphaCount (const char* str, int* result) // считает, сколько букв в строке

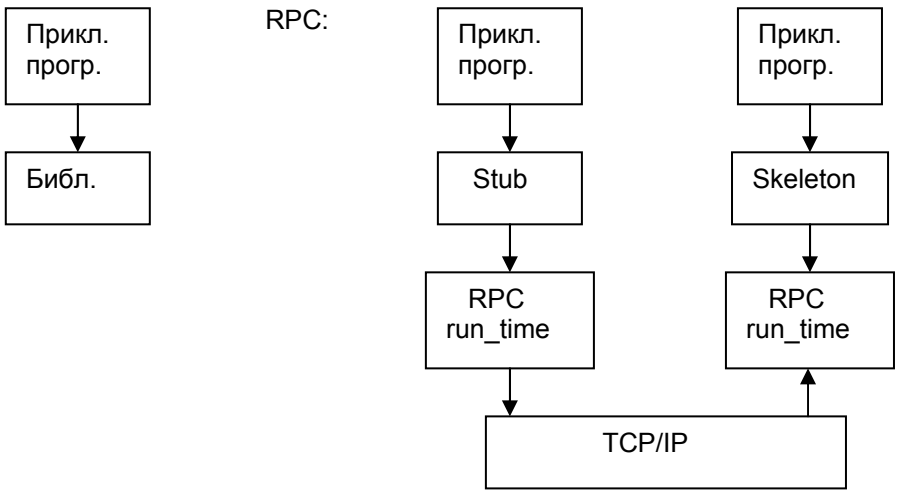
.cpp {

…

}

Для библиотеки на основе описания прототипа функции создаются специальные переходники.

Схема взаимодействия:



Stub – переходник, с которым взаимодействует локальная программа. Его интерфейс полностью повторяет интерфейс библиотеки таким образом, что для прикладной программы вызов абсолютно прозрачен. Но вместо того, чтобы выполнять действия, которые выполняет библиотека, он выполняет marshaling параметров процедур (передача параметров из адресного пространства прикладной программы в адресное пространство библиотеки через границы компьютеров). Это делается следующим образом: устанавливается TCP-соединение с переходником Skeleton, который работает в качестве сервера – принимает через TCP-соединение входные параметры, их интерпретирует, восстанавливает параметры из потока и вызывает функцию библиотеки. Понятно, что каждый Stub и каждый Skeleton специфичен для библиотеки. Когда функция в библиотеки заканчивает работу, она возвращает управление в Skeleton, который делает следующее:

Защищённые информационные системы. Технология “Эльбрус”

Технология “Эльбрус” зародилась в конце 60-х начале 70-х прошлого столетия. Первый компьютер, который заработал по этой технологии, заработал в 1978г. В 1985г заработал “Эльбрус - 2”, и в 1996г. – “Эльбрус - 3”.

В 2003-2004 – завершена разработка “Эльбрус – 3М” и начат её выпуск. Используя технологию “Эльбрус” работают системы ПВО, атомные электростанции.

Основные принципы:

Эльбрус – компьютер, у которого Ассемблер является высокоуровневым и использование этого Ассемблера не нарушает защищённости программ. Если рассматривать программы на языках высокого уровня, то они являются защищёнными (на Oberon при компиляции есть проверка недопустимых операторов, Java и C# - также защищённые, но не полностью).

При использовании ссылок программист не нарушает защищённость, но при реализации концепции происходит нарушение типизации, следовательно, может быть нарушение защищённости. Тот Ассемблер, в который превращается код, будет незащищённым. Используя этот Ассемблер можно взламывать программы (с помощью использования различных регистров).

При создании “Эльбрус” разработчики взяли строго типизированные языки высокого уровня и попытались создать Ассемблер, где поддерживается защита модулей друг от друга. Для этого было сделано следующее:

1) если в памяти есть ссылки, то эти ссылки нельзя интерпретировать как переменные других типов и переменные нессылочных типов нельзя интерпретировать как ссылки. Это достигается благодаря тэгированию ОП. Если говорить о 32-х разрядных компьютерах, то там тегируются каждые 32 бита. На каждые 32 бита используются 2 дополнительных бита, которыми помечается, хранится в этих 32-х разрядах ссылка или данные (а 2 бита, т.к. ссылки могут быть разных видов). Эти 2 бита, кажется, требуют изменение ОП, но эти 2 бита могут быть задействованы из битов четности. Т.е. компьютеры по технологии Эльбрус можно строить с использованием обычных устройств памяти

2) операция создания ссылки и уничтожения – это операция Asm-ра, т.е. операторы new и delete – это операторы Asm-ра. Создание ссылки – это выделение для нее области памяти, причем эта область памяти должна быть чиста (там не должно быть других ссылок). Там должны быть такие значения, чтобы обращение к этой области памяти было исключительной ситуацией, которая возникнет при обращении к памяти.

Вместо процессора – исполняющее устройство. Вместо ОЗУ – информационное устройство. Все вычисления разбиваются на модули – ВМ – 1-й защищенности.

Информационная система – работа обрабатывающего устройства, которое считывает данные в информационное пространство. При этом действия в информационном пространстве могут меняться.

Вычислительный модуль можно понимать как какую-то подпрограмму. Он характеризуется 3-мя величинами:

1) глобальные данные
2) параметры
3) локальные данные

Эти составляющие образуют контекст вычислительного модуля.

В защищенной системе должен быть статический контроль типов данных. Можно аппаратно помечать в памяти, но это не эффективно. Поэтому в памяти помечаются тегами только ссылки. Ссылки могут быть различных разрядов: 32, 64, 128. Со ссылкой работа всегда идет целиком (нельзя взять половину ссылки). Так как ссылки различных разрядов, то для тегирования памяти выделяется 2 дополнительных бита (4 значения):

1) обычное значение
2) ссылки 32 бита
3) ссылки 64 бита
4) ссылки 128 бит

Все данные, доступные по ссылке, называются контекстом.

Контексты могут быть непересекающимися, вложенными и частично непересекающимися.

Пример: пересекающиеся контексты: две вложенные подпрограммы в третьей.

```
var
  D: integer;
procedure A(x, y: integer);
procedure B;                    // B и C имеют доступ к X и Y,
```

```
…                                   // т.е. контексты B и C пересекаются
end;
procedure C;
…
end;
…
end;
```

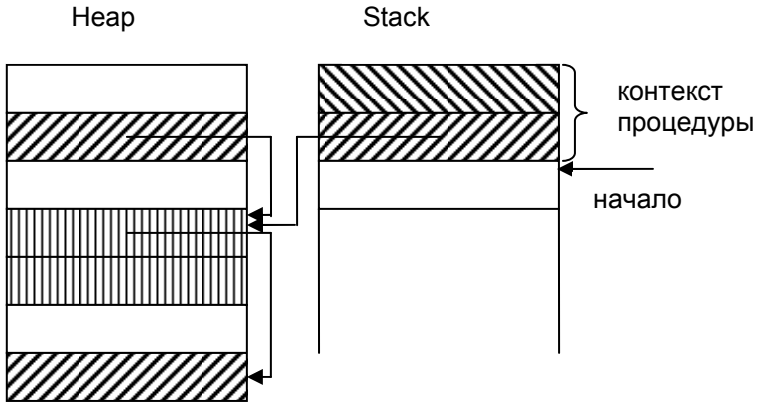
Используется контекстная защита: если у программы есть ссылка на контекст, то внутри него программа может делать все, кроме выхода за границы этого контекста.

Проблема: при вызове процедур необходимо выполнить переключение контекста.

Вместо атрибутов доступа делается следующая передача прав процедуре, выполняющейся при передаче ссылки. Ссылка, через которую доступны все остальные ссылки – контекстная ссылка.

Метка вычислительного модуля – специальная контекстная ссылка на код, для которой запрещены операции пересылки, а разрешена операция вызова с передачей управления и операция возврата.

Наименьшим элементарным звеном защиты является контекст. Контексты защищены друг от друга и от закономерного использования программ, которые работают в других контекстах.



Если в программе есть ссылка, то она передается в качестве аргумента в стеке в какую-то процедуру. Процедура размещает в стеке еще свои локальные переменные.

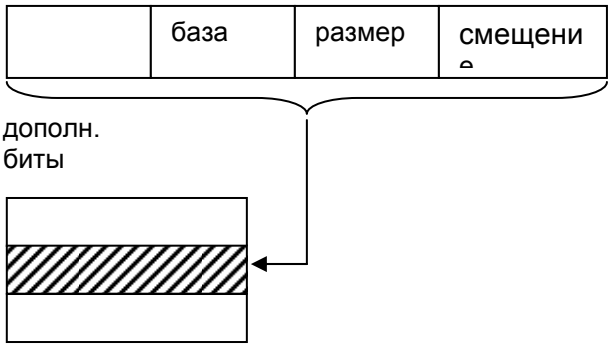
Контекстная ссылка хранит информацию о начале и о длине блока, в пределах которого действует контекст.

Доступ к заштрихованной области контролируется аппаратно. Всегда базово-индексная адресация.

Если на блок имеется ссылка, то после удаления блока обращение по этой ссылке будет приводить к прерыванию. Память считается бесконечной (C++). При удалении – заполняется пустыми значениями. Если out of memory – компрессия.

Разрядность дескрипторов (или ссылок) в различных поколениях Эльбрус – разная. В Эльбрус-3 – 128 бит; предыдущая – 64 бита.

Структура 128-битного: первые три 32-х разрядных слова – это специальная ссылка (специальный теговый бит), последнее 32-х разрядное слово не помечено. Первые три слова – база, размер блока; младшие 32 бита – смещение, позиция в блоке.



Дополнительные биты помечают права доступа.

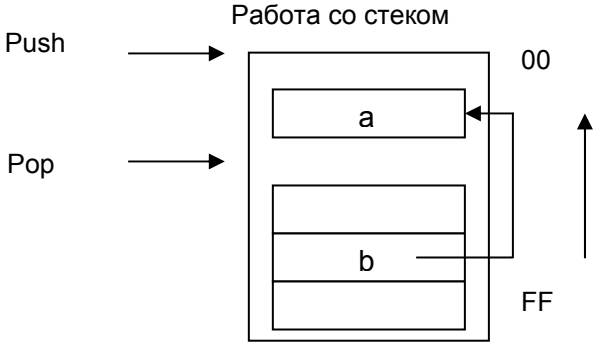
При обращении – всегда база + смещение проверит, что смещение не выходит за размер. Допускается модификация смещения как целого числа (инкремент, декремент).

В Intel защита: есть сегментные регистры, программе выделяется виртуальный адрес, по этому указателю располагается таблица (глобальная таблица дескрипторов). При использовании этого указателя проводится проверка атрибутов доступа. Таким образом, указатель на данные не несет информацию о правах доступа, о размере, т.е. программе можно сформировать любой адрес.

В Эльбрус программа не может создать недопустимый указатель (аппаратный уровень).

Второе отличие – в Intel контекст защиты – это виртуальное адресное пространство программ. Переключение контекстов происходит лишь при переключении задач, т.е. сегментных регистров (глобальных сегментных регистров).

В Эльбрус вызов из одной подпрограммы другой – это переключение контекстов.



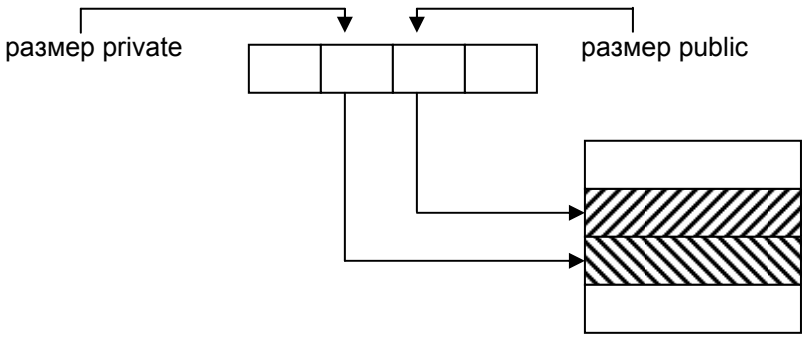
При вызове процедуры были помечены адреса:

```
int * Do          // на стеке объявляется переменная, но т.к она локальная,
                  // то из контекста она выпадает
{
   int a = …
   return & a;
}

void Do2( )
{
  int *b = Do( );          // указатель указывает на указатель стека
  *b+ = 1;
}

void Do3( )
{
   int c = 5;
}
```

Если в *void Do2( )* ставим вызов *int *b = Do3( ),* в *a* вписывается *c*. По указателю *b* лежит другое значение. Так делать нельзя. Это семантическая ошибка.

В Эльбрусе это решается так: указателям, которые находятся в стеке, разрешено указывать только в одну сторону – в начало стека.

Для поддержки ООП в Эльбрус введены специальные механизмы: дескрипторы (указатели ссылки)- 128 битные. В них не просто размер области данных, в них два размера: public-области, private-области. Т.е. чтобы из внешних процедур нельзя было модифицировать приватную часть.



Если метод внутри объекта (private), то во время работы регистр тега типа объекта содержит этот объект. Таким образом, аппаратно можно отличать private-части объекта и public.

Защищенная файловая система

Полная защищенность без защищенности другой программы не возможна. Например, одна программа открывает файл и портит его.

В UNIX – права выдаются пользователю.

В Эльбрус права выдаются программистам. Таким образом, если программа авторизована делать какие-то действия, то она другого не сделает. Нет необходимости переключаться в режим системного администратора для запуска утилит. Причем, если программе передан маршрут к файлу или какой-то каталог, то считается, что если передан файл, то переданы права на его использование.

В язык для высокого уровня на Эльбрусе введены два типа данных:

1) имя файла;

2) ссылка на файл.

Если программе (подпрограмме) передаётся ссылка на файл, то в ссылке определены права (чтение/запись, как открыт файл). Это специальный тип данных, контролируемый аппаратно. Для имен файлов – специальный тип данных, который отличается от строки. Благодаря этому, при компиляции программы информация о всех используемых каталогах и именах собирается в отдельную системную таблицу, которая помещается в заголовок .exe файла. Этот набор каталогов и файлов на диске образует контекст программы во внешней памяти (или внешний контекст программы). Попытка программы обратиться за границы этого контекста вызывает аппаратно-диагностическое прерывание.

Если ссылка, то мы не получим ссылку на файл, если контекст этого не допускает. В .exe прописывают файлы, если нужен доступ к системным файлам.

Прогноз: новое качество, которое заставит потребителей купить новые системы – это защищенные системы, т.е. компьютеры, построенные по технологии Эльбрус.

Сравнительная характеристика технологий .Net и Java.

1. Технология Java изначально разрабатывалась для встраивания в переносимые устройства, и лишь потом была преобразована для переносимого ПО в настольных компьютерах, позднее для серверного ПО. .Net – сразу для настольных компьютеров и серверов.
2. Обе технологии используют понятие промежуточного кода (для Java – Java-байт код, .Net – IL)
3. Обе среды используют автоматическое управление памятью.
4. Обе среды используют средства безопасности и контроля, которые позволяют использовать в Internet.

Основное преимущество .Net – более высокая продуктивность во всех сферах (более высокая производительность программного кода):

а) более высокоуровневый промежуточный язык. Промежуточный Java-байт код был разработан для создания виртуальных Java-машин. Java-байт код является упрощением всех команд процессора, а модель виртуальной Java-машины – пересечение всех архитектур. В виртуальной Java-машине ограничено количество ресурсов (их 4):

1)на стек

2)на динамическую память

3)на текущую исполняемую инструкцию

4)вспомогательный регистр

Промежуточный язык IL среды .Net принципиально был разработан не для интерпретации, а для динамической компиляции.

Пример программы на промежуточном языке IL, который стандартизирован, универсальный, платформонезависимый Assembler:

. assembly exern mscorlib {} // будут использоваться сборки (assembly), внешняя
. assembly hello {} // mscorlib
. method static public void main ( ) cil managed // 3

```
{   . entrypoint
    . maxstack 1
     ldstr "Hello world!"
     call void [mscorlib] System.Console:: WriteLine (class System.String)
     ret
}
```

3 – объявляется статическая функция *main* с атрибутом *cil* (common intermediate language), *managed* – код может инспектироваться во время работы

entrypoint – точка входа в функцию

maxstack – сколько элементов стека будет использоваться

ldstr – загружает на стек (исполняющие системы Java и **.**Net являются стековыми) указатель на строку

далее вызывается функция *WriteLine* для вывода этой строки на экран

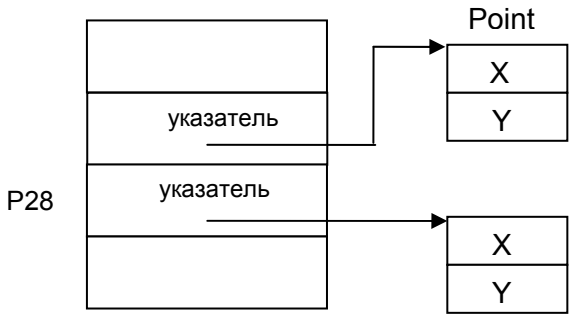
ret – выход из процедуры

Для виртуальных методов – директива virtual.

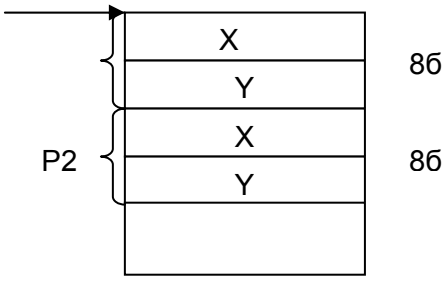
Когда запускается программа, то исполняющая система загружает промежуточный код и преобразовывает в двоичную систему.

В промежуточном коде операция сложения двух чисел в Assembler – функция add (она одинакова для сложения целых и вещественных чисел).

б) в Java нет деления на размерные типы и ссылочные, а в **.**Net – есть. Следовательно, для платформы Java стек организован следующим образом: все элементы стека имеют фиксированный размер (8 байт). При этом, в этом элементе хранится признак типа (либо это число, либо это указатель), или само значение, или ссылка на объект. Из-за этого при передаче параметров возьмет в качестве параметра функцию рисования прямоугольника (принимает две точки). Элементом стека такой объект быть не может, этот объект выделяется в динамической памяти (heap).



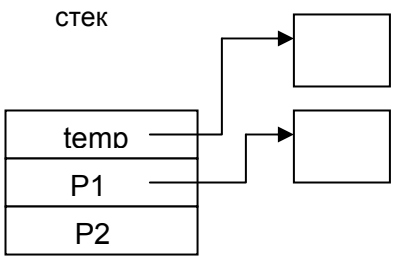
Передача Point в **.**Net: Point в **.**Net является не ссылочным типом, а размерным. Следовательно, он может передаваться следующим образом:



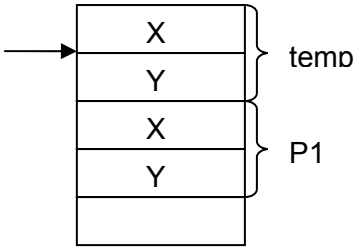
Но если объект достаточно большой, его выгодней передавать через указатели (как в Java).

```
void DrawPoint (Point p1, Point p2)
{
   Point temp = new Point (p1);
}
```

Для Java



Для .Net



5) среда Java является многоязыковой, .Net - полиязыковой. Это обеспечивается благодаря тому, что в .Net стандартизирован промежуточный язык IL; в .Net разработана общая система типов (CTS), стандартизированы базовые и структурные типы данных, стандартизированы представления данных. Существует стандарт CLS (Common Language Specification) – набор правил, типов данных для межъязыковых вызовов (для доступа к программам, написанных на многих языках). Согласно CLS, программы могут обмениваться только ограниченным набором данных (если целые числа, то только знаковые).

CLS – исполняющая система (вызывает динамический компилятор, базовую библиотеку, сборщик мусора).

В Java нет межъязыкового взаимодействия.

6) отличия в организации взаимодействия с подпрограммами, написанными на Assembler целевого процессора (на C, C++, Assembler). В Java это организуется так: виртуальная Java-машина предоставляет API JNI-набор процедур прикладного программирования, который получает, например, по символическому имени адрес процедур, находит по имени какой-то объект или адрес параметра на стеке по его имени. Требуется специальное программирование, чтобы организовать вызов dll-библиотеки из программы, написанной на Java. Этот вызов работает медленно.

В .Net это решено по-другому: средства вызова разных процедур встроены в саму вызывающую систему, преобразование параметров осуществляется автоматически. Вызов из dll-библиотеки быстрее.

Общее для Java и .Net.

Используется хранение метаданных в исполняемых модулях. Информация об используемых в программе типах данных. Это надо для:

1) работы сборщика мусора
2) обеспечения безопасности программ
3) обеспечение высокоскоростной компиляции программ

Отличия метаданных .Net от метаданных Java: в Java нет непосредственного доступа к метаданным, а в .Net программист может использовать метаданные. Сами метаданные в .Net представлены в виде обычных объектов самой платформы (для C# метаданные представлены в виде объектов языка C#. Например, в базовом классе Object есть метод Object.Type( ), который возвращает объект, описывающий тип).

В Java есть простые и ссылочные типы данных. Свои простые типы создавать нельзя, а ссылочные – можно. В .Net есть ссылочные и размерные типы. Также есть простые типы, которые определены в самой платформе (ссылочный базовый тип – string). Программист сам может определять размерные типы данных.

Средства удалённого вызова .NET Remoting.

```
class Point
{
   void SetX(int v);
   void SetY(int v);
   void SetZ(int v);
}
```

Если класс Point находится на удалённом компьютере, то установка X,Y,Z должны быть в одном методе

SetXYZ(int x, int y, int z);

т.к. каждый отдельный метод – это отдельный запрос => нагрузка на трафик.

.NET Remoting – средство удалённого вызова процедур у объекта. Эта технология позволяет создавать классы с набором методов и средств, обращаться к ним. Сначала просто программируем класс, не думая о том, что он будет работать удалённо.

Пример:

```
using System;                          // это класс
using System.Runtime.Remoting;
namespace Samples
{
  public class SampleObject:MarshalByRefObject
  {
     public SampleObject( ) { }
     public string GetString( )
     {
        return "Hello";
     }
  }
}
```

```
using System;                          // это сервер
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels.Tcp;
namespace Samples
{
   public class SampleServer
   {
      public static int Main(string[ ] args)
      {
     (1)  TcpChannel channel = new TcpChannel(10000);
     (2)  ChannelServices.RegisterChannel(channel);
     (3)  RemotingConfiguration.RegisterWellKnownServiceType ( typeof(SampleObject),
                                    "SampleService", WellKnownObjectMode.SingleCall);
          Console.WriteLine("Press Enter to exit");
          Console.ReadLine( );
      }
   }
}
```

Если создаём ссылку на удалённый объект, то возвращается ссылка на так называемый "proxy", который преобразует методы и свойства объекта к передаче по сети, приёму результатов и возврат управления пользовательской программе.

Из-за того, что клиенту возвращается ссылка на "proxy-объект" (который занимается переадресацией), удалённые объекты должны наследоваться от MarshalByRefObject. В нашем классе реализуется метод GetString.

Сервер:

(1) создание канала, через который будет выполняться передача информации (есть TcpChannel и HttpChannel для использования соответствующих протоколов). В скобках - номер порта.

(2) регистрация канала в системе (канал добавляется в глобальную службу). Этот канал вводится в общий пул каналов.

(3) регистрация в системе класса SampleObject (т.е. класс объекта, который будет удалённо использоваться)

При создании сервера не конструируется объект. Есть конструирование объекта, если клиент произвёл запрос (этот параметр WellKnownObjectMode.SingleCall)

```
using System;                          // это клиент
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Tcp;
public class SampleClient
{
   public static int Main(string[ ] args)
  {
(1)  TcpChannel channel = new TcpChannel(10000);
(2)  ChannelServices.RegisterChannel(channel);
(3)  SampleObject obj = (SampleObject)Activator.GetObject(typeof(SampleObject),
                                                  "tcp://localhost:10000/SampleService");
     if(!obj.Equals(null)) Console.WriteLine(obj.GetString( ));
  }
}
```

Работа механизма удалённого вызова методов объектов построена на следующем: когда клиент запрашивает ссылку на удалённый объект, то ему возвращается не ссылка на экземпляр локального объекта, а ссылка на так называемый "proxy - объект", который генерируется автоматически системой по типу. Эта генерация возможна благодаря мета-информации. Задача "proxy - объекта" – обеспечение передачи параметров по сети через соответствующий канал связи на удалённый компьютер.

Работа клиента состоит из:

(1) создание объекта канала для организации связи (у нас TCP, можно http и ещё свои каналы создавать, если реализованы свои протоколы).

(2) регистрация канала в системе .ChannelServices – класс, в котором статические методы. Логически для клиента представляет интерес сам удалённый объект, а не протокол.

(3) у стандартного класса Activator вызываем GetObject (т.е. получаем ссылку на удалённый объект). Туда передаётся информация о типе удалённого объекта (используя которую будет создан "proxy - объект" и настроен). Плюс передаётся строка - глобальный уникальный идентификатор объекта в сети (там: протокол (tcp), имя в сети (localhost) - можно имя или tcp-адрес, далее – порт, потом имя удалённого объекта (имя службы)).

Теперь будет создан "proxy - объект", будет установлено соединение с удалённым объектом.

Когда пропадает ссылка на объект, то объект будет уничтожен.